Università Politecnica delle Marche

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica e dell'Automazione

Tesi di Laurea

Progettazione e implementazione di un'app Android a supporto delle attività di trekking nella Regione Marche

Design and implementation of an Android app to support trekking activities in Marche Region

Relatore

Prof. Domenico Ursino

Candidato

Michele Pasqualini

Anno accademico 2019-2020

Indice

Elenco delle figure

Elenco dei listati

Introduzione

Negli ultimi anni si è assistito ad uno sviluppo tecnologico molto importante il quale ha spostato l'attenzione su apparecchiature mobili capaci di offrire ottime prestazioni a dimensioni contenute. Tale sviluppo ha permesso la nascita dei primi smartphone, il cui successo fu immediato a tal punto che, al giorno d'oggi, chiunque ne possiede almeno uno. Nei primi anni gli smartphone non erano dotati di molte funzionalità, sia perchè le risorse hardware presentavano ancora molti limiti sia perchè programmare questo tipo di dispositivi era abbastanza complicato. Tali limiti, però, furono superati in pochissimo tempo; si è passati da una situazione iniziale in cui gli smartphone potevano eseguire poche applicazioni limitate, allo scenario odierno, dove praticamente esiste un'app per qualsiasi cosa: modificare le foto con degli effetti, segnare degli eventi su un calendario, ascoltare la musica, gestire le carte fedeltà dei negozi, etc. Infatti, questi dispositivi sono ormai paragonabili a un computer di fascia medio-alta e, nei prossimi anni si assisterà sicuramente a uno sviluppo sempre più intenso, che permetterà di realizzare dispositivi autonomi con grande potenza di calcolo in grado di eseguire operazioni complesse.

Parallelamente allo sviluppo degli smartphone è, quindi, aumentato lo sviluppo della programmazione mobile, reso possibile anche dal fatto che ognuno può realizzare la propria app. Realizzare un'app partendo da zero non è del tutto banale; infatti, bisogna conoscere come opera l'ambiente di sviluppo ed avere familiarità con i componenti che esso mette a disposizione, ma, soprattutto, si devono possedere nozioni su un linguaggio di programmazione (ad esempio, Java per Android) generalmente orientato agli oggetti. Questo processo tecnologico è tuttora in evoluzione e lo possiamo vedere con le numerose versioni di Android, per il quale molto frequentemente, ne viene rilasciata una nuova.

Android è il sistema operativo sviluppato da Google per i dispositivi mobili. Si tratta di un software realizzato per funzionare soltanto su alcuni dispositivi e, quindi, non modificabile dagli utenti. Tutte le versioni di Android sono state sviluppate in Java, su sistemi di tipo embedded Linux.

Nonostante il cuore del software sia vincolato, in realtà gran parte del codice è disponibile liberamente nella modalità open source. In questo modo molte aziende possono personalizzare il sistema operativo, in base alle esigenze dei propri dispositivi. Oggi Android è un software affidabile, efficiente ed estremamente versatile, con un'ampia scelta di app scaricabili gratuitamente o a pagamento da Google Play

Store. La presenza di licenze aperte, anche dette open source, ha reso possibile un grande proliferare di applicazioni, che rappresentano un vantaggio per tutti: gli utenti possono avere a disposizione diversi servizi sul proprio dispositivo, mentre le aziende sviluppano app a costi più o meno contenuti.

Il mercato delle applicazioni mobile è cresciuto a dismisura, specialmente quello delle app dedicate ad attività sportive. Infatti, oggi, le applicazioni dedite a tali attività sono moltissime e vanno da una semplice fitness app ad un'app più articolata utile allo svolgimento dei vari sport, come il trekking. Questo mercato è molto florido, e diventa sempre più popolare con il passare del tempo; esso risulta, infatti, così innovativo e di punta che alcune aziende hanno incrementato la produzione di questa tipologia di applicazioni.

La presente tesi si colloca in tale contesto applicativo. Infatti, si è voluto realizzare un'applicazione che fosse semplice, intuitiva e che utilizzasse poche risorse dello smartphone; si è, quindi, optato per realizzare un'app dedicata ad una attività sportiva, quella del trekking. L'applicazione è stata sviluppata per documentare e semplificare l'attività all'utente. Il progetto prevede la realizzazione di una base di dati che contiene tutti i percorsi disponibili all'interno dell'app che l'utente può visionare, una serie di attrazioni e prodotti relativi sempre agli itinerari, le recensioni sugli itinerari, e una serie di percorsi creati dagli utenti stessi. Per la realizzazione dell'app è stata effettuata dapprima la progettazione dello schema entità-relazione e, poi, quella dello schema relazionale. Successivamente, si è passati alla progettazione dell'app vera e propria, mostrandone, attraverso dei diagrammi UML e dei mock-up, le funzionalità che essa dovrà fornire. L'implementazione è stata realizzata mediante l'IDE Android Studio e, nella sezione specifica della presente tesi, verranno riportati i listati raffiguranti le istruzioni di codice relative alle varie parti dell'app.

L'applicazione, sebbene risulti essere molto semplice vista dall'esterno, è dotata di una complessità implementativa non indifferente; infatti, essa deve gestire la connessione con il database, il login, la registrazione di un utente e la visualizzazione della mappa associata ad un percorso. Inoltre, l'app deve anche visualizzare le attrazioni ed i prodotti tipici legati al percorso che viene selezionato dall'utente. Riassumendo, l'applicazione ha un aspetto semplice ed abbastanza intuitivo, ma al suo interno gestisce molte funzionalità. Inoltre, se si vuole essere al passo con i tempi, in un'applicazione l'aspetto grafico risulta essere fondamentale, soprattutto se l'applicazione è abbastanza articolata. Di conseguenza, anche l'app oggetto della presente tesi deve essere dotata di una grafica colorata ed accattivante, che incentivi l'utente ad utilizzarla.

La presente tesi è strutturata come di seguito specificato:

- Nel Capitolo 1 si parlerà di Android, della sua storia, del suo successo, della struttura e delle caratteristiche di un'app per tale sistema operativo. Dopo aver introdotto Android, verranno descritte le diverse tipologie di app attualmente presenti e saranno illustrati i pro e i contro delle stesse, indicando le motivazioni che hanno spinto la scelta di una tipologia, piuttosto che di un'altra, per la realizzazione del presente progetto. Infine, si parlerà degli strumenti messi a disposizione dei programmatori per realizzare app native.
- Nel Capitolo 2 verrà effettuata una descrizione del contesto, in particolare sul trekking, per poi trattare l'analisi dei requisiti, funzionali e non funzionali.

Insieme ad essi saranno presenti i diagrammi dei casi d'uso.

- Il Capitolo 3 verterà sulla progettazione. Per quanto riguarda la base di dati, verranno mostrati gli schemi entità-relazione e relazionale, mentre, per la progettazione dell'app, verranno presentati il diagramma delle attività, per comprendere il funzionamento dell'applicazione, e i mock-up, per avere un'idea del suo aspetto.
- Il Capitolo 4 mostrerà l'implementazione del progetto, analizzando tutti i passaggi da effettuare facendo riferimento ai listati di codice. Successivamente verrà riportato il manuale utente, cioè una guida che illustra tutti i passaggi che un utente generico deve effettuare per poter utilizzare l'applicazione, specificandone le funzionalità principali.
- Nel Capitolo 5 verrà presentata una discussione in merito al lavoro svolto, indicando le nozioni che si sono apprese nel corso della realizzazione del progetto e illustrando Open Street Map, in modo da poterne comprendere i vantaggi e gli svantaggi.
- Infine, il Capitolo 6 riporta le conclusioni del progetto realizzato, indicando possibili miglioramenti e sviluppi futuri.

1

Android

In questo capitolo verranno presentate una panoramica della storia di Android con tutte le sue versioni, nonchè la struttura di un'app insieme alle sue caratteristiche e alle diverse tipologie esistenti. Infine verrà indicato quali sono gli strumenti di sviluppo per la realizzazione di app.

1.1 Introduzione ad Android

Android (Figura 1.1) è il principale sistema operativo mobile al mondo, il quale, secondo le statistiche, anima circa tre dispositivi su quattro, lasciando il restante spazio ad IOS di Apple. Ad Android è stata attribuita la più veloce diffusione mai vista per un sistema operativo mobile, infatti, il suo punto di forza riguarda il fatto che è open-source. Pertanto chiunque lo desideri, può accedere al codice, ed eventualmente, personalizzarlo.

Figura 1.1. Logo attuale di Android Inc.

Android è stato progettato principalmente per sistemi embedded, quali smartphone e tablet, ma ad oggi è diffuso anche in applicazioni specializzate per automobili (Android Auto), orologi da polso (Wear OS) e televisori (Android TV).

1.1.1 Cenni Storici

Nel 2003 Andy Rubin, Rich Miner, Nick Sears e Chris White fondarono una società denominata Android Inc. Inizialmente Android doveva fungere da sistema opeativo per fotocamere digitali, in modo che gli utenti potessero installare nuove applicazioni ed aggiungere nuove funzionalità ai loro dispositivi. Ben presto, però, viene deciso di invertire rotta e virare sul mercato degli smartphone. Nel 2005, con l'acquisizione da parte di Google per una cifra intorno a 50 milioni di dollari, Android inizia a diventare un sistema operativo con un kernel Linux. Nello stesso anno della presentazione del primo iPhone, Google annuncia la costituzione dell'Open Handset Alliace (OHA), un consorzio che riuniva una lunga lista di produttori di smartphone e telefonini. L'obiettivo era quello di gettare le basi per lo sviluppo di standard aperti in ambito mobile. Nel 2008 HTC lancia il modello Dream; si tratta del primo smartphone Android della storia. Nel 2009 arriva la prima release ufficiale di Android, ovvero Android 1.5 Cupcake e, nello stesso anno, Android si attesta al 2,9% del mercato internazionale iniziando a rosicchiare quote di mercato a IOS, Symbian e tutti gli altri. In poco tempo vengono rilasciate altre due versioni del sistema operativo: Android 1.6 Donut e Android 2.9 Eclaire. Da questo momento in avanti le versioni rilasciate saranno molte e con nomi simili a quelli di dolci.

1.1.2 Versioni

Le versioni esistenti di Android (Figura 1.2) sono diverse e, generalmente, ne viene rilasciata una ogni anno circa. Esse adottano un particolare sistema di naming: si segue l'ordine alfabetico e, in base alla lettera, viene attribuito un nome di un dolce alla versione rilasciata. Le versioni finora rilasciate sono:

- Android 1.0, API 1: 2008;
- Android 1.1, API 2: 2009;
- Android 1.5, Cupcake: 2009;
- Android 1.6, Donut: 2009;
- Android 2.0-2.1, Eclair: 2009;
- Android 2.2-2.2.3, Froyo: 2010;
- Android 2.3-2.3.7, Gingerbread: 2010;
- Android 3.0-3.2.6, Honeycomb: 2011;
- Android 4.0-4.0.4, Ice Cream Sandwich: 2011;
- Android 4.1-4.3.1, Jelly Bean: 2012;
- Android 4.4-4.4.4, KitKat: 2013 ;
- Android 5.0-5.1.1, Lollipop: 2014;
- Android 6.0-6.0.1, Marshmallow: 2015 ;
- Android 7.0-7.1.2, Nougat: 2016;
- Android 8.0-8.1, Oreo: 2017 ;
- Android 9.0, Pie: 2018;
- Android 10, Q: 2019.

È importante ricordare che, in commercio, esistono molte versioni differenti contemporaneamente, e che, in genere, un'applicazione deve essere in grado di funzionare sul numero di versioni più ampio possibile in modo tale che possa essere utilizzata

Figura 1.2. Loghi delle versioni di Android in ordine di uscita

dalla maggior parte degli utenti. Un'app lavora sulla versione per cui è stata progettata e su tutte le versioni successive. Le versioni più recenti hanno sicuramente maggiori strumenti a disposizione, ma potranno lavorare su un numero limitato di dispositivi. Quelle meno recenti, invece, perderanno in termini di prestazioni, ma, al contempo, saranno accessibili a più utenti.

1.2 Struttura di un'app Android

La parte più visibile di Android è il suo sistema operativo il quale si pone tra utente e hardware. Un sistema operativo svolge tre funzioni:

- gestisce l'hardware per conto delle applicazioni;
- fornisce servizi ad applicazioni, come il networking, la sicurezza e la gestione della memoria;
- gestisce l'esecuzione delle applicazioni.

Per descrivere l'architettura di Android ci aiutiamo con la Figura 1.3, la quale ci consente di mettere in evidenza i componenti principali organizzati secondo una struttura a layer. Abbiamo:

- *Kernel Linux*: contiene l'implementazione di una serie di driver di interazione con l'hardware, per l'utilizzo, ad esempio, della memoria o della batteria. È il layer di competenza dei vari costruttori di dispositivi, che dovranno creare i driver per il proprio hardware, in modo da sfruttarne al massimo le caratteristiche e le potenzialità.
- *Hardware Abstraction Layer* (HAL): è un livello di programmazione che consente a un sistema operativo di interagire con un dispositivo hardware a livello generale o astratto, anziché a livello hardware dettagliato tramite delle librerie. L'HAL può essere chiamato dal kernel del sistema operativo o da un driver del dispositivo.
- *Librerie Native*: sono le librerie fondamentali che consentono di gestire il software. Esse gestiscono diverse cose, come, per esempio, i browser (WebKit), la grafica (OpenGl Es) o i database.
- *Android Runtime*: rappresenta l'ambiente dove vengono eseguite le applicazioni. A differenza di altri programmi Java, gli eseguibili Android sono file `.dex` e sono destinati ad essere eseguiti su una macchina virtuale ottimizzata per dispositivi mobile a bassa potenza.

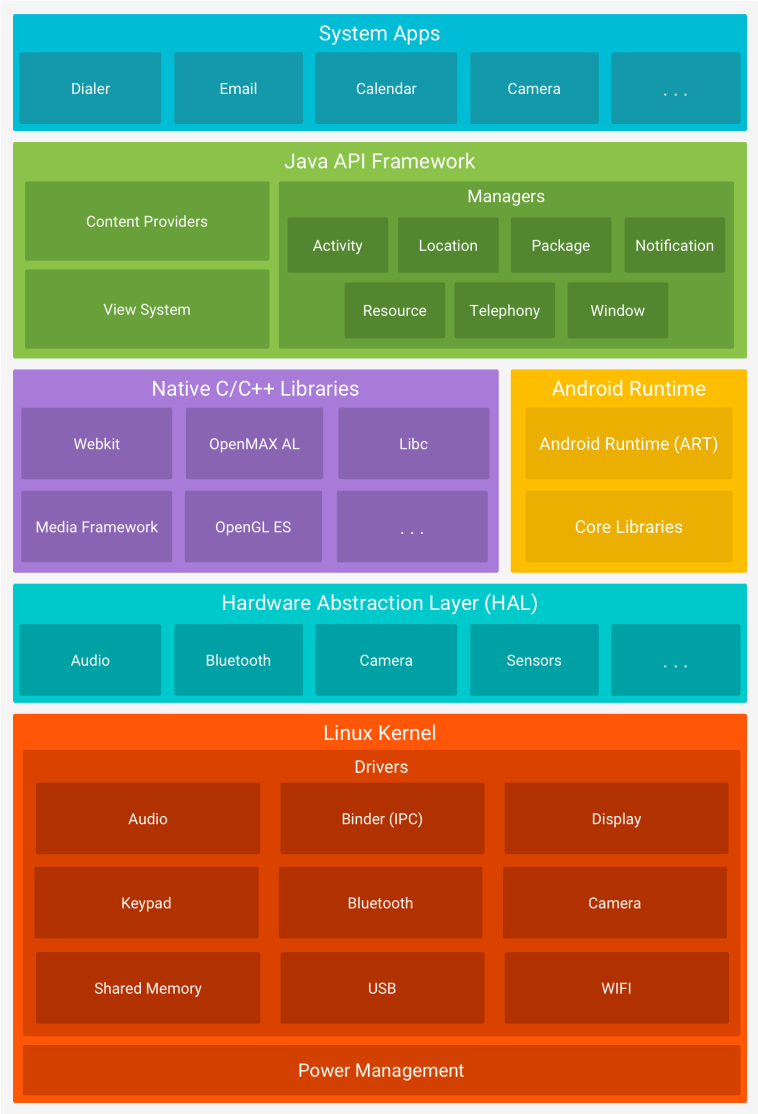

Figura 1.3. Architettura di Android

- *Java Api Framework*: si tratta del livello che contiene tutte le librerie che ci occorrono per scrivere applicazioni. Tale strato ha il compito di controllare il ciclo di vita delle applicazioni (Activity Manager), permettere alle diverse applicazioni di scambiarsi dati fra loro (Content Provider), permettere di mostrare notifiche all'utente (Notifications Manager), mettere a disposizione una serie di viste utilizzabili per creare le interfacce utente (View System), gestire i package (Package Manager), gestire i servizi telefonici (Telephony Manager), gestire i servizi di geolocalizzazione e permettere alle applicazioni di ricevere aggiornamenti riguardanti la posizione (Location Manager), e altro.

- *Application*: è il livello più in alto dove risiedono tutte le nostre applicazioni, sia quelle che scriviamo che quelle pre-costruite con il sistema operativo. Diversamente da quelle di sistema, le applicazioni utente possono essere installate e disinstallate senza che siano necessari permessi particolari.

1.3 Caratteristiche di un'app Android

Android è, quindi, una piattaforma nella quale vengono eseguiti alcuni componenti che costituiscono i fondamenti delle nostre applicazioni. La caratteristica principale di un componente è, sicuramente, quella di essere riutilizzabile, ma anche il fatto di possedere un ciclo di vita che regola le sue interazioni.

1.3.1 Activity

Con il termine "activity" ci si riferisce semplicemente ad una schermata, che è quella che vediamo quando azioniamo il nostro smartphone. Ciascuna applicazione è costituita da schermate che permettono non solo la visualizzazione delle informazioni, ma anche l'inserimento delle stesse. Ogni schermata, definisce principalmente due aspetti: l'insieme degli elementi grafici e la modalità di interazione con essi. Le regole di layout relative ad un'activity vengono definite attraverso opportune righe di codice su documenti XML, sfruttando alcuni strumenti messi a disposizione dall'IDE. Un'activity avrà, quindi, la responsabilità di gestione dei componenti dell'interfaccia utente e le interazioni con i servizi di gestione dei dati.

1.3.2 Ciclo di vita di un'activity

Dal momento in cui viene lanciata l'applicazione, fino al momento in cui essa viene messa in background, l'activity passa attraverso varie fasi che costituiscono il suo ciclo di vita.

Per definire un'activity è necessario che la nostra classe estenda la super classe `Activity`, il quale mette a disposizione una serie di metodi per gestire tutti gli eventi che ne governano il ciclo di vita. La Figura 1.4 mostra gli eventi associati:

- `onCreate()`: viene eseguito quando l'activity viene invocata; ha lo scopo di definire gli elementi di base per il funzionamento.
- `onStart()`: corrisponde al momento in cui l'activity è visibile all'utente.
- `onResume()`: questo metodo viene chiamato quando l'activity è pronta ad interagire con l'utente.
- `onPause()`: si tratta del momento in cui l'activity corrente viene messa in pausa, in favore dell'apertura di un'altra activity.
- `onStop()`: corrisponde a quando l'activity non è piu visibile all'utente.
- `onRestart()`: l'activity sta per essere riavviata dopo che è stata precedentemente arrestata.
- `onDestroy()`: permette di svolgere le ultime operazioni prima che l'app venga terminata.

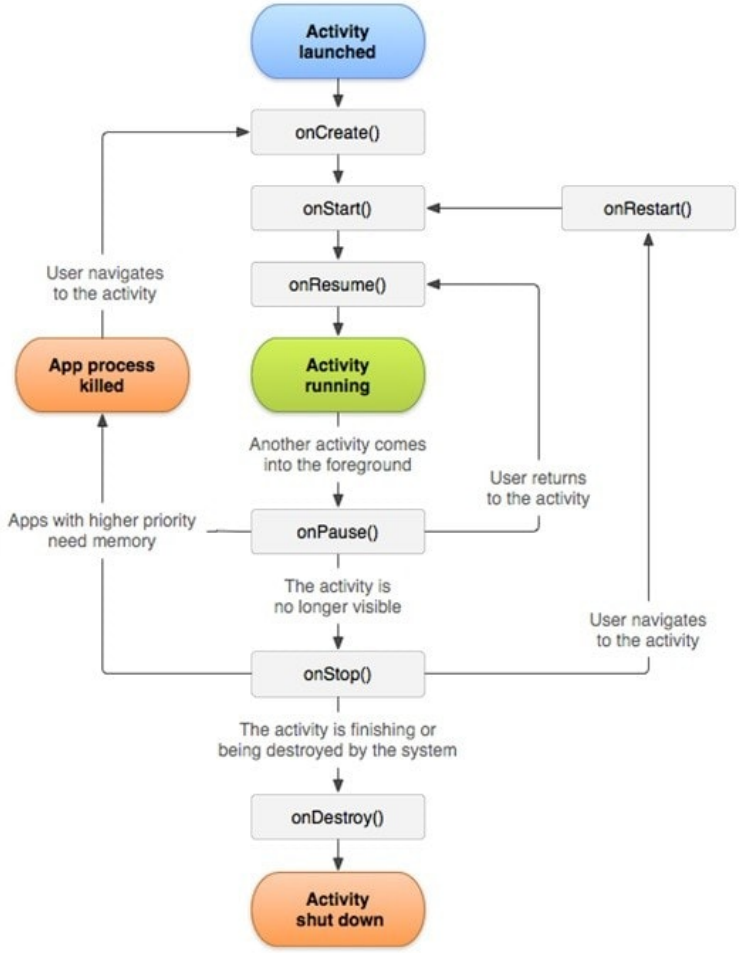

Figura 1.4. Ciclo di vita di un'activity

1.4 Tipologie di app

In informatica, un'applicazione mobile, o semplicemente app, è un'applicazione software dedicata a dispositivi mobile, quali smartphone o tablet. Un'app può essere di tre tipi (Figura 1.5):

- nativa;
- web;
- ibrida.

Ogni categoria ha i propri vantaggi e svantaggi; di conseguenza, non esiste un unico modo per realizzare app, ma, in base alle esigenze, può essere più conveniente usare una tipologia rispetto ad un'altra.

Figura 1.5. Le diverse tipologie di App

1.4.1 App native

Le app native sono delle applicazioni sviluppate specificatamente per un determinato sistema operativo mobile. I sistemi operativi più diffusi sul mercato sono IOS e Android (Figura 1.6). Le App Native adottano un linguaggio di programmazione differente da un sistema operativo all'altro: nel caso di IOS, vengono utilizzati linguaggi Objective-C e Swift; invece nel caso di Android, si utilizza Java.

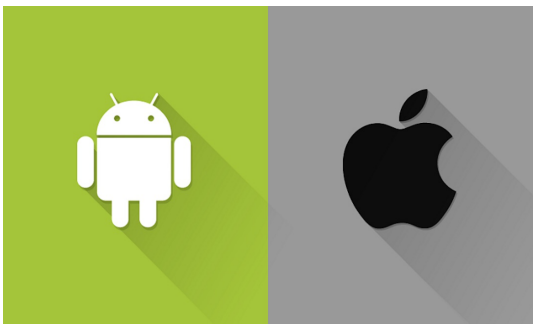

Figura 1.6. I principali sistemi operativi Mobile

I vantaggi principali di un'app nativa sono i seguenti:

- velocità, affidabilità e migliore reattività; queste sono tutte caratteristiche che hanno l'obiettivo di migliorare l'esperienza utente;
- accesso all'hardware e al software installato nel dispositivo (come fotocamera, microfono, bluetooth, rubrica, etc);
- notifiche push che permettono di avvisare gli utenti;
- funzionamento offline.

Quindi, un'app nativa non garantisce solo ottime prestazioni e il pieno sfruttamento delle funzionalità del telefono, ma si adatta a ciascuna piattaforma per

consentire agli utenti un'esperienza ideale. Il principale svantaggio delle app native riguarda la portabilità; ciò significa che è necessario distribuire una versione dell'app nativa su ogni sistema operativo attualmente in commercio.

1.4.2 Web app

Le web app sono delle applicazioni che possiedono le stesse funzionalità di un sito web, senza nessuna differenza tra piattaforma, sistema di sviluppo e codice. Ciò significa che gli utenti non devono installare l'applicazione sul dispositivo, ma l'app verrà eseguita tramite un browser e, quindi, non inciderà sulla capacità di memoria del telefono. La realizzazione di questa tipologia di applicazione è legata a strumenti di sviluppo classici dei siti web, ovvero linguaggi di mark-up (HTML5 e CSS3), linguaggi di programmazione lato client (Javascript) e lato server (PHP). Le web app presentano diversi vantaggi; in particolare:

- funzionando come un sito web, possono girare su tutti i dispositivi allo stesso modo, senza alcuna differenza a livello di codice;
- non devono essere sottoposte al processo di approvazione del Market;
- non è necessaria l'installazione, quindi esse consentono di risparmiare la memoria del dispositivo;
- lavorano con un numero di risorse inferiore rispetto ad un'app normale.

Le web app si distinguono molto dalle app native, ma dobbiamo tenere conto anche dei vari svantaggi che le caratterizzano; in particolare:

- in termini di prestazioni, non possono sfruttare al massimo le funzionalità del dispositivo e, quindi, risultano essere più lente;
- necessitano di una connessione ad Internet per operare;
- non vengono scaricate, quindi si ha una bassa fidelizzazione.

Va evidenziato che gli obiettivi per cui generalmente viene costruita una web app sono ben diversi dagli obiettivi che caratterizzano un'app nativa. Nel caso della prima lo scopo è quello di rendere disponibile un contenuto al maggior numero possibile di utenti, mentre, nel caso della seconda, lo scopo è quello di fornire servizi cercando di sfruttare al massimo le capacità dello smartphone.

1.4.3 App ibride

L'anello mancante tra le due tipologie di app descritte sono le app ibride, chiamate anche multipiattaforma. Le app ibride sono una via di mezzo tra le app native e le web app. Esse, infatti, sono in grado di girare su sistemi operativi differenti utilizzando lo stesso codice sorgente (o almeno una buona parte). Queste applicazioni sono composte da una parte cross-platform e da una parte specializzata, cioè specifica per il sistema operativo su cui devono funzionare. Questo aspetto garantisce un enorme vantaggio in quanto si riesce ad ottenere la portabilità tipica delle web app avendo le performance di un'app nativa. Rispetto alle app native, le app ibride sono più rapide da sviluppare e meno dispendiose. Un altro importante vantaggio dello sviluppo ibrido sta nel fatto che viene generata una sola versione dell'applicativo,

quindi gli aggiornamenti e l'aggiunta di nuove feature sono più semplici ed efficaci. Tuttavia, tutti questi benefici hanno un costo: le app ibride sono più lente delle app native; questo perchè tali app si devono poter adattare ad una vasta gamma di dispositivi diversi.

In generale, abbiamo visto i diversi tipi di applicazioni con i loro punti di forza e le loro debolezze. Quando si sviluppa un'app la scelta della tipologia viene effettuata in base ad una serie di fattori, come i requisiti del progetto, il target di riferimento, gli obiettivi preposti e il budget che si vuole investire. Quindi possiamo affermare che, in base a cosa dovrà fare l'app, sarà più conveniente usare una tipologia rispetto ad un'altra.

1.5 Strumenti di sviluppo

Android Studio (Figura 1.7) è l'ambiente di sviluppo integrato (IDE) ufficiale per lo sviluppo di app Android, basato su IntelliJ IDEA di Jetbrain. È stato annunciato il 16 maggio 2013 in occasione della conferenza Google I/O. Prima di Android Studio programmare app Android non era così immediato. Nelle prime versioni era presente un kit di sviluppo, rilasciato nel 2008, composto da una serie di strumenti a riga di comando e script predefiniti, e per questo abbastanza difficile da utilizzare e poco versatile per la maggior parte degli sviluppatori.

Figura 1.7. Logo dell'IDE Android Studio

Oltre al potente editor di testi e agli strumenti di sviluppo di IntelliJ, Android Studio offre ancora più funzionalità che migliorano la produttività durante la creazione di un'app Android. Esso, infatti, offre un sistema di generazione flessibile basato su Gradle, un emulatore veloce e ricco di funzioni, un ambiente unificato in cui è possibile sviluppare per tutti i dispositivi, Android Studio, inoltre, è in grado di applicare modifiche al codice e alle risorse dell'app in esecuzione senza riavviarla, è possibile, infine, integrare Android Studio con Git Hub per lo sviluppo di un progetto complesso in team.

Ogni progetto in Android Studio contiene uno o più moduli con file di codice sorgente e file di risorse. I tipi di moduli includono:

- moduli per app Android;
- moduli per le librerie;
- moduli di Google App Engine.

Per default, Android Studio visualizza i file all'interno di un progetto nella vista Android Project, come mostrato nella Figura 1.8. Questa vista è organizzata in moduli che forniscono un rapido accesso ai file chiave del progetto.

I componenti fondamentali sono:

- *Manifest*: contiene il file `AndroidManifest.xml`; esso definisce il nome dell'app, quale schermata verrà mostrata per prima quando l'utente apre l'app, cosa può fare l'applicazione, a che tipo di richieste può rispondere, se può andare su Internet, i permessi di cui necessita, etc.
- *Java*: contiene i file del codice sorgente Java, ovvero tutti quei file che implementano l'app.
- *Res*: contiene le immagini e tutti i layout appartenenti all'applicazione, cioè una serie di file XML che, attraverso opportune righe di codice, definiscono la struttura di un'activity.
- *Gradle*: è un sistema basato sulle idee di Apache Ant e Apache Maven utilizzato per dichiarare la configurazione di un progetto.

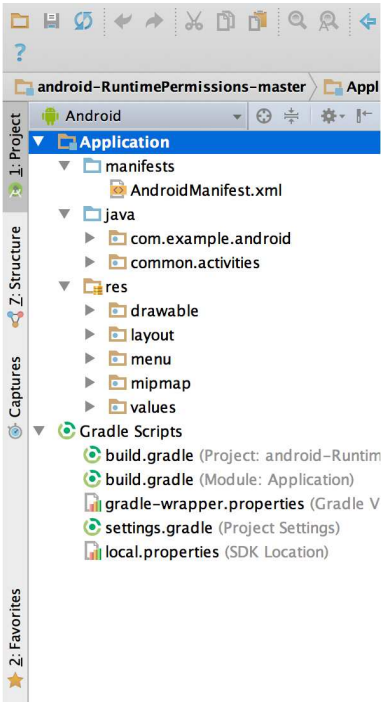

Figura 1.8. Organizzazione file in un progetto Android

2

Descrizione del contesto e analisi dei requisiti

In questo capitolo verranno illustrate descrizione del contesto e analisi dei requisiti. In particolare, verranno introdotti l'attività di trekking, specialmente nella Regione Marche, ed i benefici che da essa ne derivano. Infine, verranno effettuate un'analisi dei requisiti ed una rappresentazione del caso di studio tramite i diagrammi dei casi d'uso.

2.1 L'attività del trekking

L'attività di trekking è strettamente legata all'escursionismo; infatti il termine "trekking" deriva dal verbo inglese "to trek", che significa fare un viaggio lungo, camminando piano. Un mix tra camminata in mezzo alla natura ed escursionismo. L'escursionismo è un'attività motoria e sportiva, che consiste nel camminare lungo percorsi più o meno agevoli i quali tipicamente non prevedono la presenza di mezzi di trasporto convenzionali. Sempre più persone che si avvicinano a questa attività prediligono zone montuose o immerse nel verde, piuttosto che zone urbane o molto affollate. Tra i vari tipi di trekking si può distinguere quello itinerante e quello residenziale. Il primo, prevede lo spostarsi di giorno in giorno da una tappa all'altra, pernottando in luoghi diversi. Il secondo, invece, si contraddistingue per la scoperta di nuovi sentieri e percorsi, ritornando, però, sempre nello stesso punto di partenza verso sera. Normalmente, non è necessaria una preparazione fisica estrema; questo infatti dipende dalla tipologia e difficoltà del percorso. Se si decide di affrontare un percorso in quota, allora, in quel caso, sarà necessario procedere ad un allenamento rigoroso. Tali considerazioni vanno fatte anche per l'attrazzatura in quanto, per le prime escursioni, basteranno pochissimi oggetti, come, ad esempio, delle buone scarpe, protezione solare e dell'acqua.

2.1.1 Il trekking nella Regione Marche

La Regione Marche (Figura 2.1) sorge tra le montagne degli Appennini e il Mare Adriatico, ed è una regione molto ricca di sentieri e territori adatti al trekking. Nel patrimonio naturalistico sono presenti parchi nazionali, parchi regionali e riserve naturali davvero affascinanti.

Figura 2.1. Logo della Regione Marche.

Da Nord a Sud sono molte le attrazioni locali che, ogni anno, attirano l'attenzione di molti turisti. In particolare, tra quelli di maggior interesse troviamo:

- *Monte Conero*: il parco del Monte Conero (Figura 2.2) è un area protetta dove è possibile passeggiare in diversi sentieri che si snodano fra i boschi. Il monte ha quota massima pari a 572 metri e il suo nome deriva dalla parola greca *komaros*, ovvero "corbezzolo"; quest'ultimo è un arbusto tipico del parco che possiede rotonde bacche rosse.

Figura 2.2. Parco del Monte Conero.

- *Gola dell'Infernaccio*: esse sono delle gole naturali (Figura 2.3) scavate dal fiume Tenna all'interno del parco nazionale dei Monti Sibillini. Questo territorio è abbastanza frequentato da turisti, in quanto diversi percorsi escursionistici permettono di visitare attrazioni locali, come delle piccole cascate, la sorgente del fiume e l'Eremo di San Leonardo.

Figura 2.3. Riserva naturale della Gola dell'Infernaccio.

- *Lago di Fiastra*: il lago di Fiastra (Figura 2.4) è un lago artificiale alimentato dalle acque del fiume Fiastrone, sempre situato all'interno del parco nazionale dei Monti Sibillini. I percorsi adatti all'attività di trekking sono molti; uno, in particolare, permette ai turisti di raggiungere le famosissime Lame Rosse, delle rocce granitiche di colore, appunto, rossiccio, molto caratteristiche del territorio e soprannominate da alcuni *il canyon delle Marche*.

Figura 2.4. Lago di Fiastra.

- *Gola del Furlo*: la riserva naturale della Gola del Furlo (Figura 2.5) si trova nella provincia di Pesaro e Urbino. I sentieri che la compongono sono numerosi e tra i siti di maggior interesse troviamo la galleria romana, la grotta del grano e il fiume Candigliano che costeggia la strada. Inoltre, il sito è ideale anche per chi ama il birdwatching, viste le numerose specie di uccelli che lo frequentano.

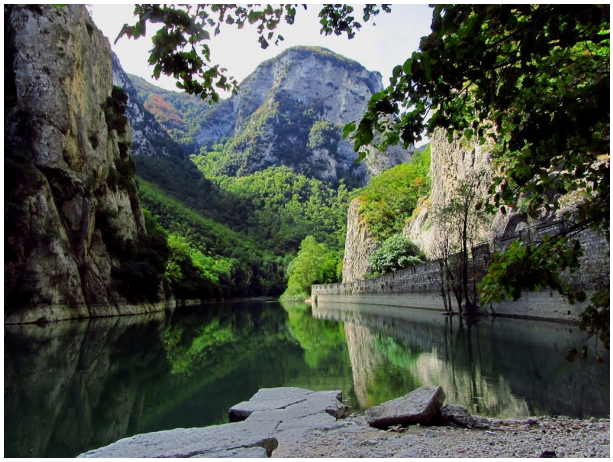

Figura 2.5. Riserva naturale della Gola del Furlo.

2.1.2 Le regole e gli effetti del trekking

Regole e responsabilità

Per vivere a pieno il trekking occorre riporre la massima attenzione nella nostra sicurezza e in quella di chi ci circonda, dall'inizio alla fine del viaggio. Durante una escursione, ognuno di noi è soggetto a dei doveri da rispettare, a delle regole a cui attenersi e ha delle responsabilità da assumere per non mettere in difficoltà se stesso e gli altri. Non si deve mai assolutamente abbandonare un percorso segnato perchè si rischia di non riuscire più ad orientarsi e di smarrire la strada del ritorno. Si deve evitare di smuovere pietre o sassi, perchè questo potrebbe risultare molto pericoloso nel caso in cui fossero presenti compagni di escursione. È di assoluta importanza rispettare l'ambiente, non gettare rifiuti di alcun tipo che possano arrecare danni e lasciare sempre puliti i luoghi dove siamo passati.

Benefici del trekking

La montagna, in generale, viene da tempo utilizzata come luogo di cura; basti pensare agli innumerevoli esempi di sanatori e terme che, nei secoli addietro, sono nati ad alte quote. L'ambiente montano non è di per sé un ambiente di cura, ma alcuni suoi elementi particolari, come l'aria o l'acqua, diventano terapeutici producendo effetti benefici sulla persona. I benefici delle camminate all'aria aperta e a contatto con paesaggi naturali sono davvero numerosi:

- modifica i comportamenti e le abitudini quotidiane;
- migliora l'autostima;
- permette di rallentare dai ritmi frenetici della nostra società e aiuta a liberarsi dalle fonti di stress;
- migliora l'apparato cardiocircolatorio;
- migliora la salute psichica;
- migliora la capacità di utilizzo dell'ossigeno quando si sale a quote elevate.

In alcuni casi, si può parlare anche di montagna-terapia, che consiste in un approccio di tipo terapeutico-riabilitativo e/o socio-educativo il quale prevede la montagna come ambiente idoneo allo svolgimento di un lavoro terapeutico orientato alla prevenzione, alla cura e alla riabilitazione di differenti problematiche.

2.2 Analisi dei requisiti

In questa sezione verrà fatta un'analisi dei requisiti, evidenziando quelli funzionali e non funzionali del progetto. I requisiti funzionali si presentano come elenchi di funzionalità o servizi che il sistema dovrà fornire. Invece, i requisiti non funzionali indicano i vincoli e le proprità relative al sistema. Dopo l'analisi dei requisiti verranno mostrati i diagrammi dei casi d'uso, che descrivono le azioni che l'applicazione deve eseguire in seguito all'interazione da parte degli utenti esterni.

2.2.1 Descrizione dei requisiti

Il progetto consiste nella realizzazione di un'app nativa Android, la quale, come obiettivo principale, ha proprio il compito di facilitare l'attività di trekking in alcuni percorsi e sentieri della Regione Marche. Lo scopo è quello di fornire all'utente che utilizza l'app una serie di informazioni che possano aiutarlo nella scelta o nella documentazione di un determinato itinerario.

Requisiti funzionali

L'applicazione deve garantire le seguenti funzionalità fondamentali:

- *Mostrare i percorsi*: l'utente deve poter visualizzare l'elenco dei percorsi presenti all'interno dell'applicazione.
- *Indicare i dettagli del percorso*: l'utente deve poter accedere alle caratteristiche dettagliate associate ad ogni percorso.
- *Visualizzare la mappa*: l'utente deve essere in grado di visualizzare la mappa di ogni singolo percorso.
- *Esporre le recensioni*: l'utente deve essere in grado di trovare le recensioni espresse sui percorsi disponibili.
- *Descrivere i punti di attrazione*: l'utente deve poter accedere alle attrazioni ed ai prodotti enogastronomici che caratterizzano un determinato percorso.

Requisiti non funzionali

L'applicazione deve rispettare i seguenti vincoli:

- *Design intuitivo*: l'utente deve essere in grado di comprendere, senza l'utilizzo di guide, come muoversi all'interno dell'applicazione per andare alla ricerca delle informazioni di cui necessita. Per questo, è necessario definire un design intuitivo e una grafica quanto più semplice possibile.
- *Gestione degli errori*: specialmente quando l'utente si trova di fronte alla sezione del login o a quella della registrazione, il sistema deve saper gestire comportamenti anomali, come il mancato completamento dei campi o la compilazione errata degli stessi, in modo tale che questi eventi non provochino un blocco o una terminazione improvvisa dell'applicazione.

2.2.2 Diagramma dei casi d'uso

La fase successiva consiste nel trovare gli attori e i casi d'uso del progetto, ovvero creare il diagramma dei casi d'uso (use case diagram). Questi sono diagrammi dedicati alla descrizione delle funzioni o dei servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono con esso. Tali diagrammi, quindi, rappresentano l'interazione dell'utente con i casi d'uso in cui esso è coinvolto. Lo scopo del diagramma è quello di individuare i casi d'uso principali senza scendere troppo nel dettaglio, cioè senza tenere conto delle sequenze di interazioni o dei dettagli implementativi.

Un diagramma dei casi d'uso è composto dal *system*, ovvero il sistema nel suo complesso, generalmente rappresentato da un rettangolo, dai *use case*, cioè le funzionalità offerte dal sistema, e dagli *actor*, cioè gli utenti che possono interagire con il sistema e richiedere l'esecuzione di alcuni casi d'uso.

Nella Figura 2.6 viene mostrato il diagramma dei casi d'uso relativo all'app da realizzare nel caso in cui l'utente sia un visiatore generico. Invece, nella Figura 2.7, viene mostrato il diagramma dei casi d'uso relativo all'app nel caso in cui l'utente sia l'amministratore.

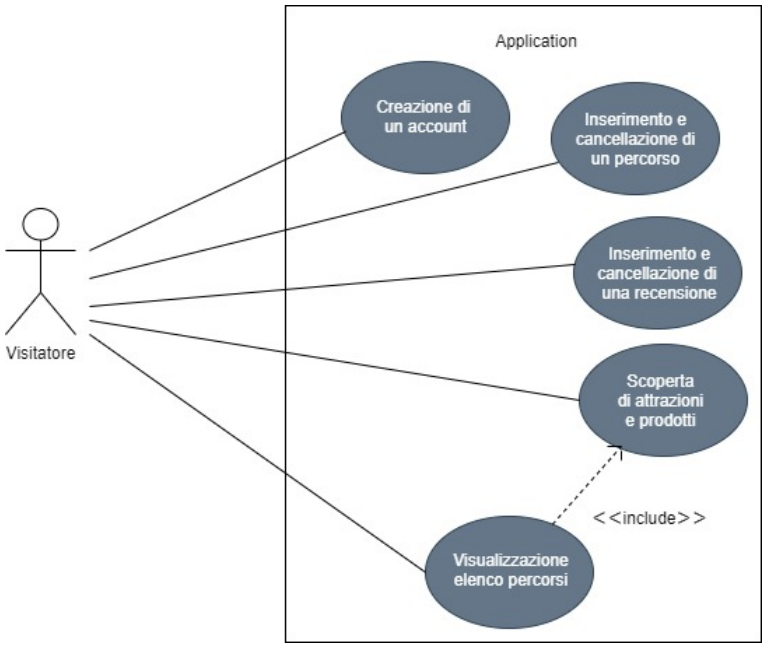

Figura 2.6. Diagramma dei casi d'uso relativo all'app nel caso in cui l'utente è un visitatore.

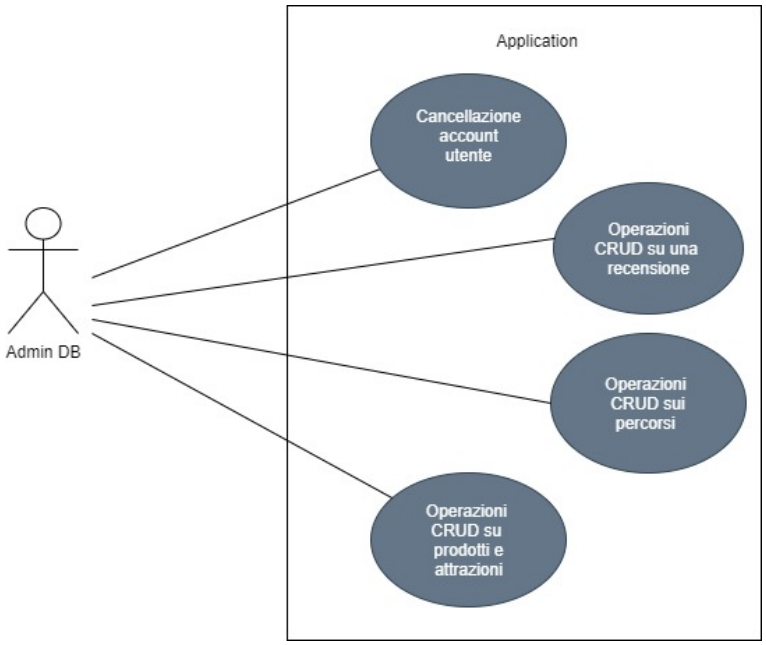

Figura 2.7. Diagramma dei casi d'uso relativo all'app nel caso in cui l'utente è un amministratore.

3

Progettazione dell'app

In questo capitolo verranno illustrate la progettazione della base di dati e la progettazione dell'applicazione. In particolare, verrano descritti lo schema entity-relationship e quello relazionale. Infine, per quanto riguarda la progettazione dell'app, verranno illustrati il diagramma delle attività e i mock-up, per dare un'idea di come dovrà funzionare l'applicazione.

3.1 Progettazione della base di dati

Lo sviluppo della nostra base di dati prevede un server di Altervista (Figura 3.1), il quale contiene il nostro database e i file `.php` che permettono all'app di interfacciarsi con esso. Questi file `.php`, a seconda della loro funzionalità, restituiranno delle conferme di avvenuto successo dell'operazione e un set di dati, in formato JSON, che verrà decodificato dall'app nelle opportune classi create all'interno del progetto. Il nostro database MySQL viene amministrato tramite phpMyAdmin, il quale consente di creare una base di dati da zero e mette a disposizione delle funzionalità per l'inserimento dei dati, per effettuare delle query e per il back-up dei dati.

Figura 3.1. Logo di Altervista.

Altervista è una spazio di hosting, o meglio, una piattaforma web, dove è possibile aprire un sito web in modo totalmente gratuito. Altervista, quindi, offre la possibilità di creare un sito web con PHP, database MySQL e accesso FTP. L'uso dello spazio è libero, ma le risorse a disposizione, come spazio web e traffico mensile, sono limitate inizialmente ed espandibili tramite l'acquisto o l'inserimento delle pubblicità. Inoltre, sono presenti applicativi auto installanti ed un site builder chiamato AlterPages.

3.1.1 Schema E-R

Nella Figura 3.2 viene mostrato lo schema *entity-relationship* associato all'app oggetto della presente tesi. Questa rappresentazione è molto utile perchè ci aiuta a tradurre le informazioni ricavate dall'analisi di un determinato dominio in uno schema concettuale. Gli elementi principali di un diagramma di questo tipo sono:

- *Entità*: rappresentano classi di oggetti (fatti, cose, persone, etc) che hanno proprietà comuni ed esistenza autonoma ai fini dell'applicazione di interesse. Generalmente vengono rappresentate mediante un rettangolo.
- *Relazioni*: dette anche associazioni, definiscono un legame tra due o più entità. Di norma, una relazione viene rappresentata graficamente da un rombo contenente il suo nome.
- *Attributi*: le entità e le relazioni possono essere descritte usando una serie di attributi. Un'attributo associa ad ogni occorrenza di una entità o di una relazione un valore, appartenente al dominio dell'attributo stesso.

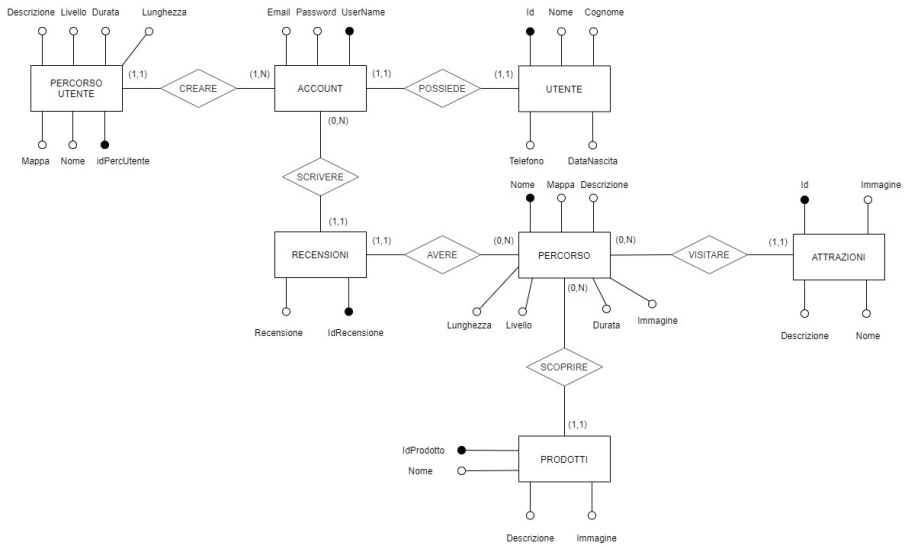

Figura 3.2. Schema E-R relativo alla base di dati del progetto.

Le entità che compongono la nostra base di dati sono:

- *Utente*: descrive le generalità di un utente (nome, cognome, telefono, etc) che intende registrarsi;
- *Account*: rappresenta un vero e proprio account associato ad un utente;
- *Percorso Utente*: sono i percorsi che vengono creati dagli utenti;
- *Recensioni*: sono i commenti lasciati dagli utenti sui percorsi;
- *Percorso*: sono gli itinerari disponibili all'interno dell'applicazione;
- *Attrazioni*: sono i punti di maggior interesse presenti in un percorso;
- *Prodotti*: sono i prodotti enogastronomici tipici legati sempre ad un itinerario.

Invece, le relazioni presenti nella nostra base di dati sono:

- *Possiede*: associa ad ogni utente l'account che possiede;
- *Creare*: associa un account ad un percorso che esso crea;
- *Scrivere*: associa le recensioni scritte dagli account;
- *Avere*: associa un percorso alle recensioni che esso possiede;
- *Visitare*: associa le attrazioni che possono essere visitate in un percorso;
- *Scoprire*: associa i prodotti tipici ad ogni percorso.

3.1.2 Schema relazione

Lo schema relazionale è un modello logico di rappresentazione dei dati di un database basato sul concetto matematico di relazione. Una relazione è sostanzialmente una tabella, dove i valori di ciascuna colonna sono omogenei tra loro e tutte le righe hanno valori diversi. Una delle fasi di maggiore importanza nella progettazione riguarda la traduzione dello schema E-R verso lo schema relazionale. Nel nostro caso, lo schema relazionale (Figura 3.3) è il seguente:

- *Utente* (Id, Nome, Cognome, DataNascita, Telefono);
- *Account* (UserName, Email, Password, IdUtente);
- *PercorsoUtente* (idPercUtente, Nome, Mappa, Livello, Lunghezza, Descrizione, Durata, userName);
- *Recensione* (IdRecensione, NomePercorso, UserName, Recensione);
- *Percorso* (Nome, Mappa, Livello, Lunghezza, Descrizione, Durata, Immagine);
- *Attrazioni* (Id, Nome, Descrizione, Immagine, NomePercorso);
- *Prodotti* (IdProdotto, Nome, Descrizione, Immagine, NomePercorso).

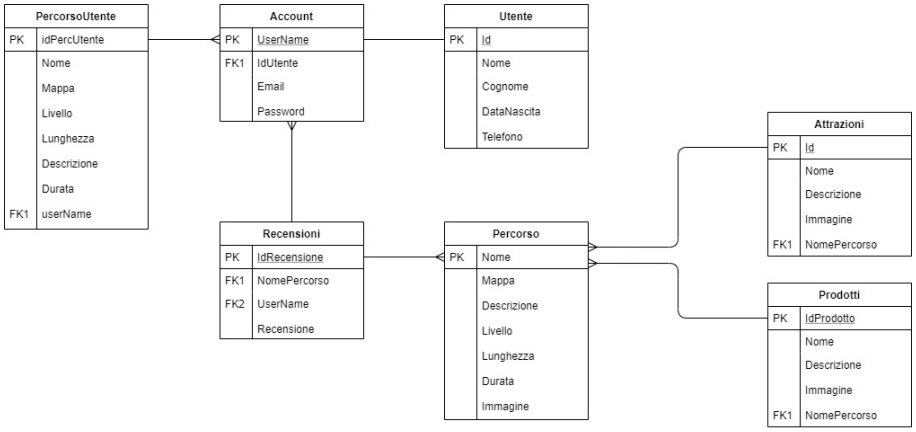

Figura 3.3. Schema logico relativo alla base di dati del progetto.

3.2 Progettazione dell'app

La fase di progettazione consiste nel progettare le funzionalità software che l'applicazione deve offrire. In questa sezione verrà costruito il diagramma delle attività e verranno realizzati i mock-up. In questo modo si vuole dare un'idea di come funzioni l'app e di quale aspetto essa debba avere.

3.2.1 Diagramma delle attività

I diagrammi delle attività sono dei diagrammi che descrivono un processo. Generalmente, i simboli grafici utilizzati sono due: i nodi per le funzionalità e gli archi per la rappresentazione dell'ordine con cui esse vengono eseguite. Questi diagrammi consistono nell'elencazione dei passaggi che devono essere effettuati per compiere una determinata azione, richiesta dall'attore. Nel caso del nostro progetto, il diagramma delle attività (Figura 3.4) illustra la sequenza di azioni che l'utente può compiere nel momento in cui inizia ad interagire con l'applicazione. Lo scopo è, quindi, quello di mostrare, in maniera ancora informale, il flusso delle azioni da compiere per raggiungere un'obbiettivo.

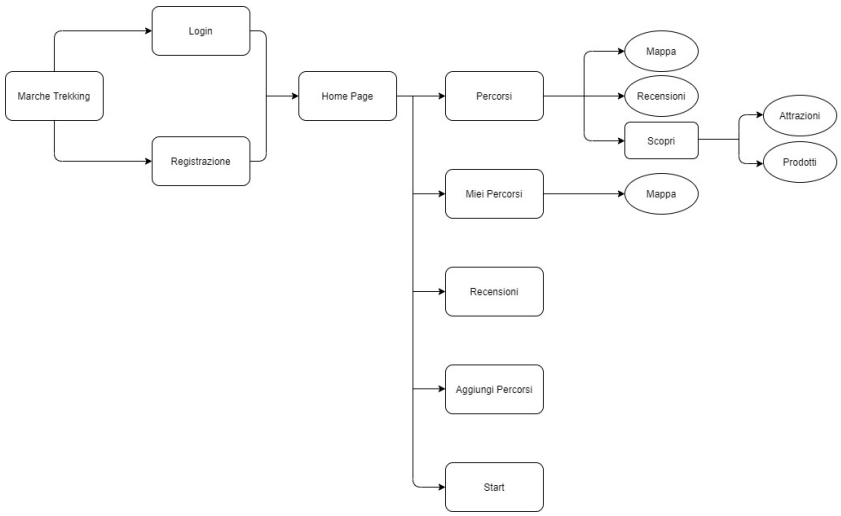

Figura 3.4. Diagramma delle attività relativo all'app.

3.2.2 Mock-up

I mock-up sono degli strumenti di progettazione i quali consistono in una rappresentazione grafica, o meglio, in uno "schizzo", del sistema senza specificare in dettaglio le funzionalità che esso offre; essi servono, quindi, per rappresentare i vari contenuti e per illustrare l'aspetto grafico del progetto. In base al grado di dettaglio, si può avere un mockup di livello 0, in cui ci si concentra sui contenuti, senza perdersi in

dettagli grafici, di livello 1, in cui si arricchisce il livello 0 con dei dettagli grafici, e, infine, di livello 2, che è in genere una "copia" del progetto finale, in cui la grafica gioca un ruolo fondamentale e si evidenziano colori e forme. Di seguito sono presentati i mockup di livello 0 per le diverse funzionalità dell'applicazione.

First activity

La first activity (Figura 3.5) è la prima pagina che viene vista dall'utente all'apertura dell'app. Essa presenta il nome e il logo dell'app, assieme a due pulsanti: Accedi e Registrati.

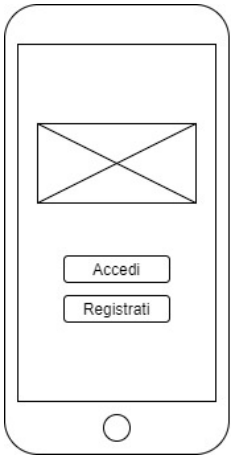

Figura 3.5. Mock-up di livello 0 riguardante la prima schermata.

Registrati

La pagina Registrati (Figura 4.5) permette all'utente di compilare una serie di campi con le sue informazioni e, quindi, di creare un account il quale dovrà essere utilizzato per poter accedere all'app.

Accedi

Digitando le proprie credenziali, ovvero username e password, l'utente può accedere ai servizi offerti dall'app (Figura 4.4).

Home page

Subito dopo avere effettuato il login, l'utente si trova di fronte alla home page dell'applicazione (Figura 4.6). Egli, tramite i pulsanti presenti, può scegliere se inserire un percorso, se consultare l'elenco degli itinerari disponibili, se scrivere una recensione, oppure se azionare un contapassi.

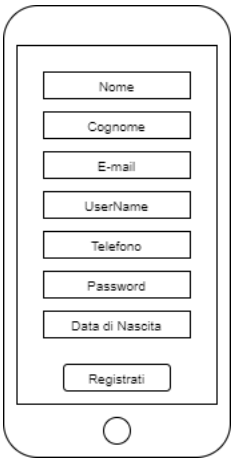

Figura 3.6. Mock-up di livello 0 riguardante la pagina di registrazione.

Figura 3.7. Mock-up di livello 0 riguardante la pagina di login.

Elenco percorsi

La Figura 3.9 mostra l'elenco di tutti gli itinerari disponibili all'interno dell'applicazione.

Dettaglio di un percorso

Questa schermata (Figura 4.8) descrive le caratteristiche dettagliate di ogni percorso e offre la possibilità di consultare la mappa, le recensioni, le attrazioni ed i prodotti tipici.

La mappa

L'utente può vedere la mappa del percorso selezionato in precedenza (Figura 4.9).

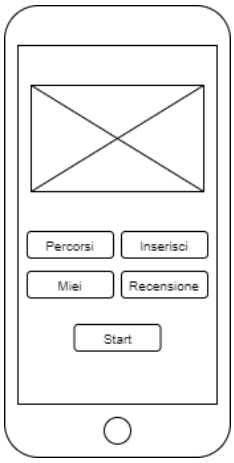

Figura 3.8. Mock-up di livello 0 riguardante la home page.

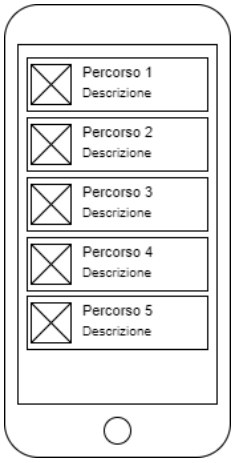

Figura 3.9. Mock-up di livello 0 riguardante l'elenco dei percorsi.

Le recensioni

L'utente può visionare le recensioni lasciate da altri utenti sempre sul percorso selezionato in precedenza (Figura 4.10).

Scopri

La pagina Scopri (Figura 4.11) pone l'utente di fronte a due scelte possibili: visionare i prodotti enogastronomici, oppure i punti di maggior interesse presenti all'interno del percorso.

Elenco prodotti tipici

La Figura 3.14 mostra l'elenco di tutti i prodotti enogastronomici appartenenti al percorso selezionato dall'utente.

Figura 3.10. Mock-up di livello 0 riguardante i dettagli di un percorso.

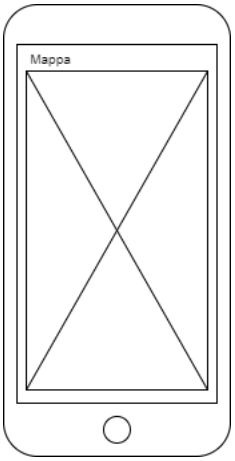

Figura 3.11. Mock-up di livello 0 riguardante la mappa di un percorso.

Elenco attrazioni locali

La Figura 3.15 mostra l'elenco di tutte le attrazioni locali presenti all'interno dell'itinerario selezionato dall'utente.

Dettaglio di una attrazione e di un prodotto

La Figura 3.16 mostra, attraverso una descrizione ed un'immagine, i dettagli associati ad un prodotto o ad un'attrazione.

Inserisci un percorso

Questa schermata (Figura 3.17) permette all'utente di inserire un nuovo percorso.

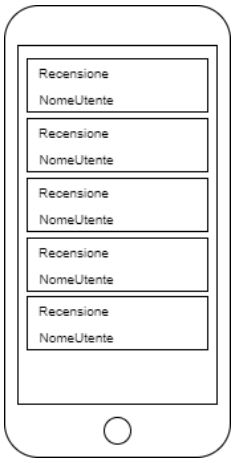

Figura 3.12. Mock-up di livello 0 riguardante le recensioni associate ad un percorso.

Figura 3.13. Mock-up di livello 0 per la visualizzazione di attrazioni e prodotti di un percorso.

Crea nuovo percorso

L'utente può creare un nuovo percorso completando tutti i campi che compaiono nell'apposita pagina (Figura 3.18).

I miei percorsi

Questa sezione (Figura 3.19) mostra tutti i percorsi aggiunti dall'utente.

Inserisci recensione

L'utente ha la possibilità di scrivere una recensione selezionando il percorso che intende recensire (Figura 3.20).

Figura 3.14. Mock-up di livello 0 relativo all'elenco dei prodotti tipici di un percorso.

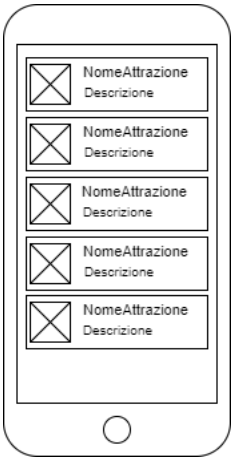

Figura 3.15. Mock-up di livello 0 relativo all'elenco delle attrazioni di un percorso.

Emergenza

La Figura 3.21 mostra la pagina dedicata ai numeri di emergenza.

Inizia attività

La Figura 3.22 mostra una sorta di registro attività, dove l'utente, oltre a visualizzare la data corrente, può azionare e fermare un contapassi.

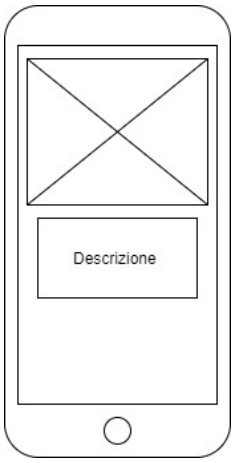

Figura 3.16. Mock-up di livello 0 relativo ai dettagli di un'attrazione o di un prodotto associati ad un determinato percorso.

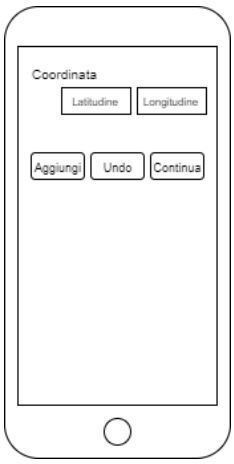

Figura 3.17. Mock-up di livello 0 riguardante l'inserimento di un nuovo percorso.

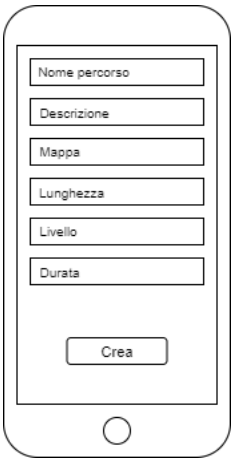

Figura 3.18. Mock-up di livello 0 riguardante la creazione di un nuovo percorso.

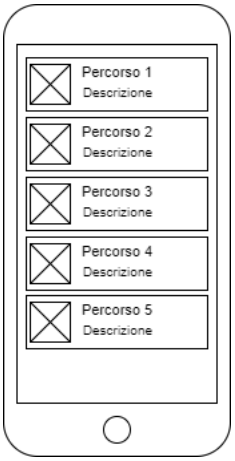

Figura 3.19. Mock-up di livello 0 riguardante l'elenco dei percorsi di un utente.

Figura 3.20. Mock-up di livello 0 riguardante l'inserimento di una recensione.

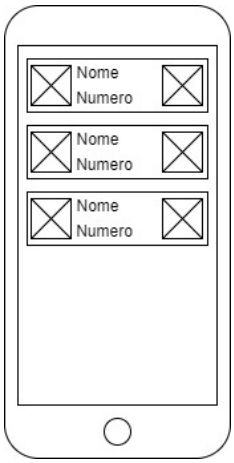

Figura 3.21. Mock-up di livello 0 relativo alla pagina contente i numeri di emergenza.

Figura 3.22. Mock-up di livello 0 relativo al contapassi.

4

Implementazione dell'app e manuale utente

Questo capitolo tratta l'implementazione dell'app descritta nel capitolo precedente. Dopo averne descritto la struttura, verranno mostrati gli script relativi all'implementazione delle varie parti del progetto. Infine, verrà mostrato il manuale utente per l'utilizzo dell'applicazione implementata.

4.1 Struttura del progetto

Per l'implementazione dell'applicazione è stata scelta, come versione minima da supportare, la versione "Android Marshmallow", che corrisponde all'API 23. Il progetto, sviluppato tramite l'IDE Android Studio, è strutturato come descritto nella Figura 4.1. Le componenti di particolare interesse, dal punto di vista implementativo, sono le seguenti:

- `manifests`: è la directory che contiene il file `AndroidManifest.xml`. Esso viene considerato come una sorta di "carta d'identità", in quanto contiene le caratteristiche basilari dell'app.
- `com.example.marchetrekking`: è la directory che contiene tutte le classi del nostro progetto; esse hanno lo scopo di implementare le funzionalità descritte nel Capitolo 3.
- `drawable`: è la directory che contiene tutte le immagini utilizzate dall'app durante il suo ciclo di vita.
- `layout`: è la directory contenente tutti i file XML; questi descrivono le modalità di visualizzazione di una schermata e degli elementi che la compongono. Ogni file XML è legato ad una particolare classe nella directory `com.example.marchetrekking`; in altre parole le classi faranno riferimento agli elementi contenuti nei rispettivi file XML, e viceversa.
- `menu`: è la directory che contiene due file XML, in particolare degli item che vengono visualizzati sulla toolbar.
- `mipmap`: è la directory che contiene l'icona dell'app, visibile dal menu di navigazione.
- `values`: è la directory che contiene dei file XML, necessari per etichettare alcune componenti fondamentali dell'app, come il nome o i colori primari e secondari.

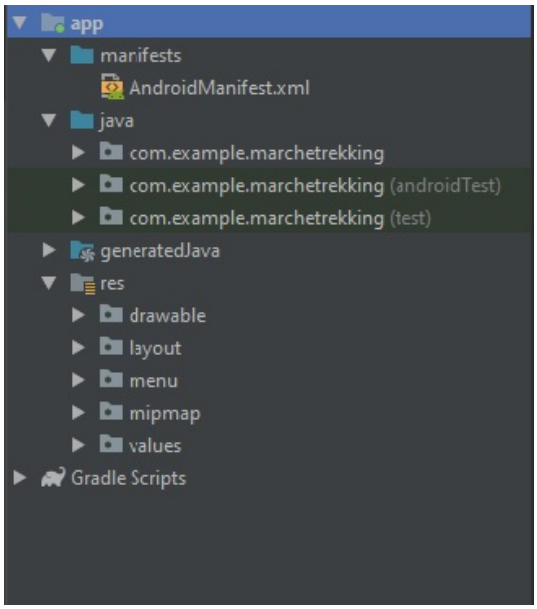

Figura 4.1. Struttura dell'applicazione.

4.2 Implementazione

In questa sezione vedremo il codice relativo ad alcune di queste componenti, facendo particolare attenzione a quelle contenute nelle directory `manifests`, `com.example.marchetrekking` e `layout`.

4.2.1 Il manifest

Come già detto in precedenza, il manifest contiene le caratteristiche principali della nostra applicazione.

Nel caso di progetto, all'interno del manifest vanno inseriti i permessi relativi ad Internet, dato che l'app per funzionare necessita di una connessione stabile. È presente, inoltre, il tag `application`, che contiene le informazioni riguardanti il nome dell'app, l'icona, il tema e le dichiarazioni delle activity del progetto (cioè le pagine che l'utente vede).

Il tag `application` è presentato nel Listato 4.1.

```
<application
        android:allowBackup="true"
        android:icon="@drawable/logo"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar"
        android:usesCleartextTraffic="true">
        <activity
            android:name=".Scopri"
            android:screenOrientation="portrait" ></activity>
        <activity
            android:name=".DettaglioProdotti"
            android:screenOrientation="portrait" />
        <activity
            android:name=".Prodotti"
```

```
            android:screenOrientation="portrait" />
        <activity
            android:name=".DettaglioAttrazione"
            android:screenOrientation="portrait" />
        <activity
            android:name=".Emergenza"
            android:screenOrientation="portrait" />
        <activity
            android:name=".Attrazioni"
            android:screenOrientation="portrait" />
        <activity
            android:name=".add_recension"
            android:screenOrientation="portrait" />
        <activity
            android:name=".Recension"
            android:screenOrientation="portrait" />
        <activity
            android:name=".Mappa"
            android:screenOrientation="portrait" />
        <activity
            android:name=".DettaglioPercorso"
            android:screenOrientation="portrait" />
        <activity
            android:name=".ElencoPercorsi"
            android:screenOrientation="portrait" />
        <activity
            android:name=".Home"
            android:screenOrientation="portrait" />
        <activity
            android:name=".aggPercorso"
            android:screenOrientation="portrait" />
        <activity
            android:name=".NuovoPercorsoDesc"
            android:screenOrientation="portrait" />
        <activity
            android:name=".Attivita"
            android:screenOrientation="portrait" />
        <activity
            android:name=".activity_login"
            android:noHistory="true"
            android:screenOrientation="portrait" />
        <activity
            android:name=".Principale"
            android:noHistory="true"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MainActivity"
            android:screenOrientation="portrait" />
    </application>
```

Listato 4.1. Il tag `application` del manifest

Si può vedere che le activity possiedono l'attributo `android:screenOrientation=''portrait''`. Quest'ultimo è necessario per impedire la rotazione dell'app in seguito alla rotazione dello schermo da parte dell'utente; tale comportamento deve essere evitato, perchè altrimenti alcuni elementi della schermata verrebbero visualizzati fuori posto, o addirittura non sarebbero visualizzati.

4.2.2 Le activity

In questa sezione verranno analizzate singolarmente le diverse activity che compongono il progetto. In particolare, si farà riferimento alla classe e al relativo file di layout XML.

Le activity del progetto sono molteplici; ognuna si occupa di implementare una funzionalità diversa. Di seguito vengono riportate quelle fondamentali:

- `MainActivity`: è l'activity che gestisce la fase di registrazione all'app da parte di un nuovo utente.

- `LoginActivity`: questa activity consente all'utente di effettuare l'accesso ai servizi dell'app.
- `HomeActivity`: è l'activity principale dove l'utente trova tutte le funzionalità di cui può usufruire.
- `ElencoPercorsiActivity`: questa activity mostra, tramite l'utilizzo di una lista, l'elenco di tutti i percorsi disponibili nell'app.
- `DettaglioPercorsoActivity`: tale activity ha il compito di illustrare una descrizione dettagliata di ogni percorso, offrendo anche la possibilità di visualizzare la mappa, le recensione e le attrazioni.
- `MappaActivity`: è l'activity che permette all'utente di visionare la mappa associata ad un percorso; quest'ultimo viene raffigurato mediante un tracciato che rappresenta le sue coordinate geografiche (latitudine e longitudine) esatte.
- `RecensioneActivity`: è l'activity dove vengono elencate le recensioni appartenenti al percorso selezionato.
- `AggiungiRecensioneActivity`: è l'activity che consente di scrivere e aggiungere una nuova recensione selezionando un percorso.
- `NuovoPercorsoActivity`: quest'activity, dopo avere completato i vari campi presenti, permette di aggiungere un nuovo percorso; esso sarà, poi, visibile dalla sezione "miei percorsi".
- `AttrazioniActivity`: quest'activity mostra, tramite l'utilizzo di una lista, l'elenco di tutte le attrazioni locali che si possono trovare in un percorso.
- `ProdottiActivity`: questa sezione illustra una serie di prodotti enogastronomici legati al percorso selezionato ed al territorio circostante.
- `DettaglioProdottiActivity` e `DettaglioAttrazioneActivity`: queste activity mostrano i dettagli di un'attrazione e un prodotto; in particolare, sono presenti un'immagine ed una piccola descrizione.

Di queste ultime quattro activity non verrà riportato il codice in quanto il meccanismo di funzionamento è simile a quello delle activity `ElencoPercorsiActivity` e `DettaglioPercorsoActivity`.

L'activity MainActivity

La `MainAcitivty` è l'activity che implementa la parte della registrazione. Il file di layout prevede una serie di EditText dove l'utente digiterà nome, cognome, email, username, telefono, password e data di nascita. Quando viene premuto il pulsante "crea account", viene aperta una connessione HTTP, in modalità POST, tra l'app e il server Altervista; quest'ultimo contiene il file `aggiungiutente.php`. Quindi, i dati verranno inviati e salvati sul nostro database. I codici del file XML e della classe sono presentati, rispettivamente, nei Listati 4.2 e 4.3.

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.marchetrekking.MainActivity"
    android:orientation="vertical">


    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbarReg"
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:background="@android:color/background_light"
    app:titleTextColor="@android:color/black"
    android:minHeight="?attr/actionBarSize"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    />


<android.support.v4.widget.NestedScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginRight="10dp"
        android:layout_marginLeft="10dp">

        <TextView
            android:id="@+id/textView3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Nome" />

        <EditText
            android:id="@+id/nome"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName" />

        <TextView
            android:id="@+id/textView5"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:text="Cognome" />

        <EditText
            android:id="@+id/cognome"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName" />

        <TextView
            android:id="@+id/textView4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:text="E-mail" />

        <EditText
            android:id="@+id/mail"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName" />

        <TextView
            android:id="@+id/textView8"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:text="UserName" />

        <EditText
            android:id="@+id/utente"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName" />

        <TextView
            android:id="@+id/textView9"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:text="Telefono" />

        <EditText
            android:id="@+id/telefono"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName" />

        <TextView
            android:id="@+id/textView11"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
```

```
            android:text="Password" />

        <EditText
            android:id="@+id/password"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPassword" />

        <TextView
            android:id="@+id/textView12"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:text="Data di nascita(AAAA-MM-GG)" />

        <EditText
            android:id="@+id/data"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
             />



        <Button
            android:id="@+id/button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dp"
            android:background="@drawable/rounded_button"
            android:text="Crea account"
            android:textColor="@android:color/white" />

    </LinearLayout>
</android.support.v4.widget.NestedScrollView>


</LinearLayout>
```

Listato 4.2. Il codice del file XML associato all'activity `MainActivity`

```
package com.example.marchetrekking;

public class MainActivity extends AppCompatActivity {

    SessionManager session;
    private String user;
    private String n;
    private String c;
    private String dnascita;
    private String telef;
    private String mail;
    private String pass;


    private EditText id, nome, cognome, data, email, password, telefono;
    private Button invia;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbarReg) ;
        toolbar.setTitle("Registrati");


        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);

        session =new SessionManager(this);
        id = (EditText) findViewById(R.id.utente);
        nome = (EditText) findViewById(R.id.nome);
        cognome = (EditText) findViewById(R.id.cognome);
        data = (EditText) findViewById(R.id.data);
        //data.setText("2005/12/31");
        telefono = (EditText) findViewById(R.id.telefono);
        email = (EditText) findViewById(R.id.mail);
        password = (EditText) findViewById(R.id.password);
        invia = (Button) findViewById(R.id.button);

        final Calendar calendar = Calendar.getInstance();
        calendar.set(2005, 11, 31);
        final SimpleDateFormat simpleDateFormat = new SimpleDateFormat( "yyyy/MM/dd", Locale.US);
        data.setText(simpleDateFormat.format(calendar.getTime()));


        final DatePickerDialog.OnDateSetListener datePicker = new DatePickerDialog.OnDateSetListener() {
```

```
    @Override
    public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
        calendar.set(Calendar.YEAR, year);
        calendar.set(Calendar.MONTH, month);
        calendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);

        data.setText(simpleDateFormat.format(calendar.getTime()));

        }
};

data.setOnFocusChangeListener(new View.OnFocusChangeListener() {
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        new DatePickerDialog(MainActivity.this,datePicker, calendar.get(Calendar.YEAR),
        calendar.get(Calendar.MONTH), calendar.get(Calendar.DAY_OF_MONTH)).show();

    }

});
data.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        new DatePickerDialog(MainActivity.this,datePicker, calendar.get(Calendar.YEAR),
        calendar.get(Calendar.MONTH), calendar.get(Calendar.DAY_OF_MONTH)).show();

    }
});

invia.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        user = id.getText().toString();
        n = nome.getText().toString();
        String nomeClean = n.replace("'", "''");
        c = cognome.getText().toString();
        String cognomeClean = c.replace("'", "''");
        dnascita = data.getText().toString();
        telef = telefono.getText().toString();
        mail = email.getText().toString();
        pass = password.getText().toString();

        String d[] = dnascita.split("/");
        if(Integer.parseInt(d[0]) >= 2005)
        {
            Toast.makeText(MainActivity.this, "Data di nascita non valida", Toast.LENGTH_SHORT).show();
            return;
        }

        if(!isNullOrEmpty(user) && !isNullOrEmpty(n) && !isNullOrEmpty(c) && !isNullOrEmpty(dnascita)
        && !isNullOrEmpty(telef) && !isNullOrEmpty(mail) &&!isNullOrEmpty(pass) ) {


            HttpURLConnection client = null;
            URL url;
            try {
                // se la richiesta è POST
                url = new URL("http://marchetrekking.altervista.org/aggiungi_utente.php");
                client = (HttpURLConnection) url.openConnection();
                client.setRequestMethod("POST");
                client.setDoOutput(true);
                client.setDoInput(true);
                // write data in request
                OutputStream out = new BufferedOutputStream(client.getOutputStream());
                BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
                String data = URLEncoder.encode("mail", "UTF-8")
                        + "=" + URLEncoder.encode(mail, "UTF-8");

                data += "&" + URLEncoder.encode("pass", "UTF-8") + "="
                        + URLEncoder.encode(pass, "UTF-8");


                data += "&" + URLEncoder.encode("n", "UTF-8") + "="
                        + URLEncoder.encode(n, "UTF-8");


                data += "&" + URLEncoder.encode("c", "UTF-8") + "="
                        + URLEncoder.encode(c, "UTF-8");


                data += "&" + URLEncoder.encode("dnascita", "UTF-8") + "="
                        + URLEncoder.encode(dnascita, "UTF-8");


                data += "&" + URLEncoder.encode("telef", "UTF-8") + "="
                        + URLEncoder.encode(telef, "UTF-8");


                data += "&" + URLEncoder.encode("user", "UTF-8") + "="
                        + URLEncoder.encode(user, "UTF-8");


                writer.write(data);
```

```
                    writer.flush();
                    writer.close();
                    out.close();

                    InputStream in = client.getInputStream();
                    String json_string = ReadResponse.readStream(in).trim();

                    if (json_string.equals("1")) {
                        session.createSession(user, pass);
                        Toast.makeText(MainActivity.this, "Inserimento effettuato", Toast.LENGTH_LONG).show();
                        Intent intent = new Intent(MainActivity.this, Home.class);
                        startActivity(intent);
                    } else {
                        Toast.makeText(MainActivity.this, "Errore nell'inserimento", Toast.LENGTH_LONG).show();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    if (client != null) {
                        client.disconnect();
                    }
                }
            }else{
                Toast.makeText(MainActivity.this, "Completa i campi",Toast.LENGTH_SHORT).show();
            }

        }

    });
    }

    public void onBackPressed() {
        Intent setIntent = new Intent(MainActivity.this, Principale.class);
        startActivity(setIntent);
        finish();
    }

    public boolean isNullOrEmpty(String s){
        return s == null || s.isEmpty();
    }
}
```

Listato 4.3. Il codice della classe associato all'activity `MainActivity`

L'activity LoginActivity

La `LoginActivity` è l'activity dove l'utente, digitando le proprie credenziali, effettua l'accesso all'app. Anche qui sono presenti due EditText, uno per lo username ed uno per la password. Tramite il pulsante "accedi", i dati verranno inviati al database e verrà controllato se quelle credenziali esistono. In caso di esito positivo, l'utente si troverà di fronte alla home dell'app. Invece, se le credenziali sono errate, oppure, se i campi sono incompleti o sono presenti caratteri non ammessi, verrà generato un Alert Dialog, il quale avviserà l'utente della presenza di un errore. I codici del file XML e della classe sono presentati, rispettivamente, nei Listati 4.4 e 4.5.

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activity_login"
    android:orientation="vertical">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbarAcc"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/background_light"
        app:titleTextColor="@android:color/black"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_gravity="center_horizontal"
```

```
        android:layout_marginTop="150dp"
        >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            >

            <TextView
                android:id="@+id/textView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="UserName" />

            <EditText
                android:id="@+id/nomeUtente"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:ems="10"
                android:inputType="textPersonName"/>
        </LinearLayout>


        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginTop="20dp">

            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Password"/>

            <EditText
                android:id="@+id/pass"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:ems="10"
                android:inputType="textPassword"/>
        </LinearLayout>



    <Button
        android:id="@+id/login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/rounded_button"
        android:text="Accedi"
        android:textColor="@android:color/white"
        android:layout_marginLeft="80dp"
        android:layout_marginTop="30dp"/>



    </LinearLayout>

</LinearLayout>
```

Listato 4.4. Il codice del file XML associato all'activity `LoginActivity`

```
package com.example.marchetrekking;

public class activity_login extends AppCompatActivity {
    SessionManager session;
    EditText u, p;
    Button login;
    TextView t;
    String ulog,plog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbarAcc) ;
        toolbar.setTitle("Accedi");

        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);

        u=(EditText) findViewById(R.id.nomeUtente);
        p=(EditText) findViewById(R.id.pass);
        login = (Button) findViewById(R.id.login);

        session =new SessionManager(this);
```

```
    ulog = u.getText().toString();
    plog = p.getText().toString();


    login.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            Log.d("Errore0", "Ok");
            ulog = u.getText().toString();
            plog = p.getText().toString();

            Pattern pattern = Pattern.compile("^[A-Za-z0-9_-]{3,20}$");
            Matcher m = pattern.matcher(ulog);

            if (!isNullOrEmpty(ulog) && !isNullOrEmpty(plog)) {
                if(!m.matches()){
                    Toast.makeText(activity_login.this, "Caratteri␣non␣ammessi", Toast.LENGTH_SHORT).show();
                }

                HttpURLConnection client = null;
                URL url;
                try {

                    url = new URL("http://marchetrekking.altervista.org/login.php");
                    client = (HttpURLConnection) url.openConnection();
                    client.setRequestMethod("POST");
                    client.setDoOutput(true);
                    client.setDoInput(true);
                    OutputStream out = new BufferedOutputStream(client.getOutputStream());
                    BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
                    String data = URLEncoder.encode("username", "UTF-8")
                            + "=" + URLEncoder.encode(ulog, "UTF-8");

                    data += "&" + URLEncoder.encode("password", "UTF-8") + "="
                            + URLEncoder.encode(plog, "UTF-8");

                    writer.write(data);
                    writer.flush();
                    writer.close();
                    out.close();

                    InputStream in = client.getInputStream();
                    String json_string = ReadResponse.readStream(in).trim();


                    if (json_string.equals("1")) {

                        session.createSession(ulog, plog);

                        Intent log = new Intent(activity_login.this, Home.class);
                        startActivity(log);

                    } else {

                        AlertDialog alertDialog = new AlertDialog.Builder(activity_login.this).create();
                        alertDialog.setTitle("Errore");
                        alertDialog.setMessage("UserName␣o␣Password␣errati!");
                        alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
                                new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int which) {
                                        dialog.dismiss();
                                    }
                                });

                        alertDialog.show();
                    }


                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    if (client != null) {
                        client.disconnect();
                    }
                }
            }else {

                Toast.makeText(activity_login.this, "Completa␣i␣campi", Toast.LENGTH_SHORT).show();
            }
        }
    });


}

public void onBackPressed() {

    Intent setIntent = new Intent(activity_login.this, Principale.class);

    startActivity(setIntent);

    return;
```

```
    }

    public boolean isNullOrEmpty(String s){

        return s == null || s.isEmpty();
    }

}
```

Listato 4.5. Il codice della classe associato all'activity `LoginActivity`

L'activity HomeActivity

La `HomeActivity` è l'activity principale della nostra applicazione. Essa presenta una serie di pulsanti che l'utente può selezionare per navigare all'interno dell'applicazione. Tali pulsanti sono azionati mediante un Intent; l'unica particolarità riguarda il pulsante "miei percorsi", in quanto nell'intent viene aggiunto un extra, che sarà poi filtrato nella richiesta HTTP per ricavare i percorsi di un'utente. Attraverso le funzionalità del SessionManager, verrà mostrato sulla Toolbar lo username dell'account che ha effettuato il login. Qui troveremo, anche, due "icon button", una per eseguire il logout e l'altra per il collegamento alla sezione emergenze. I codici del file XML e della classe sono presentati, rispettivamente, nei Listati 4.6 e 4.7.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/parent_linear_layout2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ElencoPercorsi">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbarHome"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@android:color/background_light"
            android:minHeight="?attr/actionBarSize"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:titleTextColor="@android:color/black" />

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="250dp"
            android:background="@drawable/trekking" />



    </LinearLayout>


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:orientation="vertical">

        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="2"
            android:orientation="horizontal"
```

```
            android:paddingTop="65dp"

            >

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:gravity="center_horizontal"
                android:orientation="vertical">

                <Button
                    android:id="@+id/percorsi"
                    android:layout_width="140dp"
                    android:layout_height="140dp"
                    android:background="@drawable/rounded_button_1"
                    android:text="Percorsi"
                    android:textColor="@android:color/background_light" />

                <Button
                    android:id="@+id/insPercorso"
                    android:layout_width="140dp"
                    android:layout_height="140dp"
                    android:layout_marginTop="10dp"
                    android:background="@drawable/rounded_button_3"
                    android:text="Inserisci␣\n␣Percorso"
                    android:textColor="@android:color/background_light" />

            </LinearLayout>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:gravity="center_horizontal"
                android:orientation="vertical">

                <Button
                    android:id="@+id/mypercorsi"
                    android:layout_width="140dp"
                    android:layout_height="140dp"
                    android:background="@drawable/rounded_button_2"
                    android:text="Miei␣Percorsi"
                    android:textColor="@android:color/background_light"/>

                <Button
                    android:id="@+id/fai"
                    android:layout_width="140dp"
                    android:layout_height="140dp"
                    android:layout_marginTop="10dp"
                    android:background="@drawable/rounded_button_6"
                    android:text="Recensioni"
                    android:textColor="@android:color/background_light" />

            </LinearLayout>


        </LinearLayout>

        <Button
            android:id="@+id/inizia"
            android:layout_width="match_parent"
            android:layout_height="70dp"
            android:layout_marginRight="50dp"
            android:layout_marginLeft="50dp"
            android:layout_marginBottom="60dp"
            android:text="Start"
            android:textColor="@android:color/background_light"
            android:background="@drawable/rounded_button5"/>
    </LinearLayout>



</android.support.constraint.ConstraintLayout>
```

Listato 4.6. Il codice del file XML associato all'activity `HomeActivity`

```
package com.example.marchetrekking;

public class Home extends AppCompatActivity {
    SessionManager session;
    private Toolbar t;
    Button lo;
    Button percorsi, insperc, mypercorsi, start;
    Button recension;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        session=new SessionManager(this);
        HashMap<String,String> utente = session.getUserDetail();
        String n = utente.get(SessionManager.NAME);
        String p = utente.get(SessionManager.PASS);

        t= (Toolbar)findViewById(R.id.toolbarHome);
        setSupportActionBar(t);
        getSupportActionBar().setTitle("Welcome␣" + n);

        percorsi = (Button) findViewById(R.id.percorsi);
        insperc=(Button) findViewById(R.id.insPercorso);
        recension =(Button) findViewById(R.id.fai);
        mypercorsi=(Button) findViewById(R.id.mypercorsi);
        start = (Button) findViewById(R.id.inizia);

        percorsi.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent p = new Intent(Home.this, ElencoPercorsi.class);
                startActivity(p);
            }
        });

        insperc.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent p = new Intent(Home.this, aggPercorso.class);
                startActivity(p);
            }
        });

        recension.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent p = new Intent(Home.this, add_recension.class);
                startActivity(p);
            }
        });


        mypercorsi.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent p = new Intent(Home.this, ElencoPercorsi.class);
                p.putExtra("miei", true);
                startActivity(p);

            }
        });

        start.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent p = new Intent(Home.this, Attivita.class);
                startActivity(p);
            }
        });

    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        switch (item.getItemId()){

            case R.id.l:
                session.logout();
                return true;

            case R.id.e:
                Intent p = new Intent(Home.this, Emergenza.class);
                startActivity(p);
                return true;

            default:
                return true;

        }
    }
}
```

Listato 4.7. Il codice della classe associato all'activity `HomeActivity`

L'activity ElencoPercorsiActivity

L'activity `ElencoPercorsiActivity` svolge un ruolo molto importante all'interno del nostro progetto. Essa, come prima cosa, apre una connessione HTTP con il server per estrarre i percorsi (oppure i percorsi di un utente, se nell'Intent era presente un extra). Una volta estratti i percorsi, attraverso la classe `DatiPercorsi`, dove le variabili sono i campi che costituiscono un percorso (nome, mappa, descrizione, etc), viene creato un array di percorsi. Quindi, ogni oggetto dell'array è un percorso presente nella base di dati. Infine, verrà popolata la lista, la quale consentirà all'utente di vedere nell'app un elenco ordinato degli itinerari. I codici del file XML e della classe sono presentati, rispettivamente, nei Listati 4.8 e 4.9.

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/parent_linear_layout2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ElencoPercorsi">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbarElPer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/background_light"
        app:titleTextColor="@android:color/black"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />

    <ListView
        android:id="@+id/lista"
        android:scrollbars="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:longClickable="true"/>

</LinearLayout>
```

Listato 4.8. Il codice del file XML associato all'activity `ElencoPercorsiActivity`

```
package com.example.marchetrekking;

public class ElencoPercorsi extends AppCompatActivity {

    private LinearLayout ly;
    private ListView lv;
    private ArrayList aPercorsi;

    private ArrayAdapter<String> arrayAd;
    private ArrayDatiPercorsi listAdapter;

    private Toolbar t;
    SessionManager session;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);

        session=new SessionManager(this);
        HashMap<String,String> utente = session.getUserDetail();
        setContentView(R.layout.activity_elenco_percorsi);
        t=(Toolbar) findViewById(R.id.toolbarElPer);
        setSupportActionBar(t);
        getSupportActionBar().setTitle("Percorsi");


        lv=(ListView) findViewById(R.id.lista) ;
        ly=(LinearLayout) findViewById(R.id.lypercosi) ;

        Intent i=getIntent();

        HttpURLConnection client = null;
        try {
            URL url;
```

```
            if(i.getBooleanExtra("miei", false)) {
                url = new URL("http://marchetrekking.altervista.org/myPercorsi.php?username="
                + utente.get(SessionManager.NAME));
            }else{
                url = new URL("http://marchetrekking.altervista.org/percorsi.php");
            }
            client = (HttpURLConnection) url.openConnection();
            client.setRequestMethod("GET");
            InputStream in = client.getInputStream();
            String json_string = ReadResponse.readStream(in);
            if(!json_string.equals("0[]\n")) {
                JSONObject json_data = convert2JSON(json_string);
                fill_listview(json_data, i.getBooleanExtra("miei", false));
            }
            else{
                Toast.makeText(this, "Nessuna percorso presente", Toast.LENGTH_SHORT).show();
                finish();

            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        finally{
            if (client!= null){
                client.disconnect();
            }
        }



        lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

                int a = position;
                DatiPercorsi d=listAdapter.getItem(position);
                String fe=d.getNome();
                Intent intent = new Intent(ElencoPercorsi.this, DettaglioPercorso.class);
                intent.putExtra("desc", d);
                startActivity(intent);
            }
        });

    }

    private void fill_listview(JSONObject json_data, boolean miei){

        ArrayList<DatiPercorsi> dperc=new ArrayList<>(); //creazione array di percorsi
        Iterator<String> iter = json_data.keys();//iterator che permette scorrere l'arraylist
        while (iter.hasNext()) {
            String key = iter.next();
            try {
                JSONObject value = json_data.getJSONObject(key);//valore della chiave, in questo caso nome percorso
                //prelevo i dati ottenuti in risposta dal php
                int id;
                String nome =value.getString("Nome");
                String descrizione =value.getString("Descrizione");
                String mappa=value.getString("Mappa");
                Double lunghezza = Double.parseDouble(value.getString("Lunghezza"));
                int livello =Integer.parseInt(value.getString("Livello"));
                String durata = value.getString("Durata");
                String immagine;
                DatiPercorsi dp;
                if(!miei) {
                    immagine= value.getString("Immagine");
                    //istanziazione di un oggetto per ogni percorso
                    dp =new DatiPercorsi(nome, mappa, descrizione, lunghezza, livello, durata, immagine);

                } else{
                    immagine= "logo";
                    //istanziazione di un oggetto per ogni percorso attraverso l'id
                    id = Integer.parseInt(value.getString("idPercUtente"));
                    dp=new DatiPercorsi(id, nome, mappa, descrizione, lunghezza, livello, durata, immagine);
                }


                //aggiunta dell'oggetto all'arraylist
                dperc.add(dp);
            } catch (JSONException e) {
                // Something went wrong!
            }
        }
        listAdapter = new ArrayDatiPercorsi(ElencoPercorsi.this, dperc);

        lv.setAdapter(listAdapter);

    }

    private JSONObject convert2JSON(String json_data){
        JSONObject obj = null;
        try {
            obj = new JSONObject(json_data);
            //Log.d("My App", obj.toString());
        } catch (Throwable t) {
```

```
            //Log.e("My App", "Could not parse malformed JSON: \"" + json_data + "\"");
        }
        return obj;
    }


}
```

Listato 4.9. Il codice della classe associato all'activity `ElencoPercorsiActivity`

L'activity DettaglioPercorsoActivity

Dopo aver selezionato un elemento dalla lista dei percorsi, l'utente verrà rimandato all'activity `DettaglioPercorsoActivity`, la quale illustrerà le caratteristiche fondamentali del percorso in questione. Qui, saranno presenti tre pulsanti che daranno accesso ad ulteriori dettagli. Nel caso in cui la sezione "dettaglio percorso" si riferisce ad un percorso aggiunto dall'utente, allora l'utente stesso che lo ha creato può cancellarlo. I codici del file XML e della classe sono presentati, rispettivamente, nei Listati 4.10 e 4.11.

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".DettaglioPercorso">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbarDett"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/background_light"
        app:titleTextColor="@android:color/black"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1.5"
        android:orientation="horizontal">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="match_parent"
            android:layout_height="208dp"
            android:layout_weight="1"
            app:srcCompat="@mipmap/ic_launcher" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_marginBottom="5dp">

        <TextView
            android:id="@+id/dettaglio"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="10dp"
            android:layout_marginLeft="10dp"
            android:text="TextView" />



    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:orientation="horizontal"
        android:layout_gravity="bottom"
```

```
        android:layout_marginBottom="20dp">

        <Button
            android:id="@+id/map"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:background="@drawable/rounded_button"
            android:text="Mappa"
            android:textColor="@android:color/white" />

        <Button
            android:id="@+id/recensione"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="1dp"
            android:layout_marginTop="10dp"
            android:layout_marginRight="1dp"
            android:layout_weight="1"
            android:background="@drawable/rounded_button"
            android:text="Recensione"
            android:textColor="@android:color/white" />

        <Button
            android:id="@+id/scopri"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="10dp"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:background="@drawable/rounded_button"
            android:text="Scopri"
            android:textColor="@android:color/white" />

    </LinearLayout>

</LinearLayout>
```

Listato 4.10. Il codice del file XML associato all'activity `DettaglioPercorsoActivity`

```
package com.example.marchetrekking;

public class DettaglioPercorso extends AppCompatActivity {

    private TextView t;
    private Button b;
    private Button rec;
    private String nome;
    private String mappa;
    private ImageView img;
    private DatiPercorsi dp;
    private Toolbar toolbar;
    private Button scopri;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detaglio_percorso);
        toolbar = (Toolbar) findViewById(R.id.toolbarDett) ;
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("Dettaglio␣percorso");
        t=(TextView) findViewById(R.id.dettaglio);
        b=(Button) findViewById(R.id.map);
        rec=(Button) findViewById(R.id.recensione);
        img = (ImageView) findViewById(R.id.imageView);
        scopri = (Button) findViewById(R.id.scopri);

        //prendo gli extras passato dall'intent dall'activity ElencoPercorsi
        Intent i=getIntent();
        dp = (DatiPercorsi) i.getSerializableExtra("desc");


        nome =dp.getNome();
        String descrizione =dp.getDescrizione();
        mappa=dp.getMappa();
        Double lunghezza = dp.getLunghezza();
        int livello = dp.getLivello();
        String durata = dp.getDurata();
        String immagine =dp.getImmagine();
        int drawableID = getResources().getIdentifier(immagine,"drawable",getPackageName());
        img.setImageResource(drawableID);

        t.setText("␣\n" + "Nome:␣" + nome + "␣\n"
                + "␣\n" + "Descrizione:␣" + descrizione + "\n"
                + "␣\n" + "Lunghezza:␣" + lunghezza + "km␣\n"
```

```
            + "␣\n" + "Livello␣Difficoltà:␣" + livello + "\n"
            + "␣\n" + "Durata:␣" + durata + "h.␣\n");

    b.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i= new Intent(DettaglioPercorso.this, Mappa.class);
            i.putExtra("mappa", mappa);
            startActivity(i);
        }
    });


    if(dp.getId() >= 1 ){
        rec.setVisibility(View.GONE);
        scopri.setVisibility(View.GONE);
    }else {
        rec.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(DettaglioPercorso.this, Recension.class);
                intent.putExtra("NomePercorso", nome);
                startActivity(intent);
            }
        });

        scopri.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(DettaglioPercorso.this, Scopri.class);
                intent.putExtra("NomePercorso", nome);
                startActivity(intent);
            }
        });
    }



}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_delete, menu);
    if(dp.getId() == 0){
        toolbar.getMenu().findItem(R.id.delete).setVisible(false);
    }
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

    switch (item.getItemId()){

        case R.id.delete:

            AlertDialog.Builder builder1 = new AlertDialog.Builder(DettaglioPercorso.this);
            builder1.setMessage("Cancellare␣percorso?");
            builder1.setCancelable(true);

            builder1.setPositiveButton(
                    "Yes",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {

                                HttpURLConnection client = null;
                                URL url;
                                try {
                                    url = new URL("http://marchetrekking.altervista.org/cancellaPercorso.php");

                                    client = (HttpURLConnection) url.openConnection();
                                    client.setRequestMethod("POST");
                                    client.setDoOutput(true);
                                    client.setDoInput(true);
                                    OutputStream out = new BufferedOutputStream(client.getOutputStream());
                                    BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out,
                                    "UTF-8"));

                                    String data = URLEncoder.encode("percorso", "UTF-8")
                                            + "=" + URLEncoder.encode(Integer.toString(dp.getId()) , "UTF-8");

                                    writer.write(data);
                                    writer.flush();
                                    writer.close();
                                    out.close();

                                    InputStream in = client.getInputStream();
                                    String json_string = ReadResponse.readStream(in).trim();

                                    if(!json_string.equals("ok")) {
                                        Toast.makeText(DettaglioPercorso.this, "Errore!",
                                        Toast.LENGTH_SHORT).show();
```

```
                                    }
                                    else{
                                        Toast.makeText(DettaglioPercorso.this, "Percorso cancellato",
                                        Toast.LENGTH_SHORT).show();
                                        finish();
                                        Intent p = new Intent(DettaglioPercorso.this, ElencoPercorsi.class);
                                        p.putExtra("miei", true);
                                        startActivity(p);

                                    }
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                                finally{
                                    if (client!= null){
                                        client.disconnect();
                                    }
                                }


                            }
                        });



                builder1.setNegativeButton(
                        "No",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                            }
                        });

                AlertDialog alert = builder1.create();
                alert.show();
                return true;

            default:
                return true;

        }
    }
}
```

Listato 4.11. Il codice della classe associato all'activity `DettaglioPercorsoActivity`

L'activity MappaActivity

L'activity `MappaActivity` mostra la mappa associata al percorso selezionato. Si è scelto di utilizzare Open Street Maps poiché, rispetto a Google Maps, offre una gestione delle mappe molto più semplice e più adeguata alla tipologia di percorsi presenti. Inizialmente si effettua il setting della mappa e si definiscono alcune proprietà, come il tipo di mappa utilizzata, ad esempio MAPNIK, il quale è lo standard di OSM, poi lo zoom, il multitouch control, il punto in cui la mappa verrà centrata all'avvio (tramite l'utilizzo dell'oggetto GeoPoint), e così via. L'applicazione ricava i dati prendendoli dall'oggetto Intent e li "splitta" secondo il carattere ","; esso è stato usato per separare le coordinate nel database. Vengono, poi, aggiunte alla lista di latitudini le posizioni pari e alla lista di longitudini le posizioni dispari. Una volta ricevuti tutti i punti di un determinato percorso, essi vengono memorizzati in una struttura dati. Attraverso un ciclo che scorre tutto l'array, scansioniamo tutti i punti; quest'ultimi saranno uniti tramite l'oggetto polyline il quale rappresenterà proprio il percorso. Tale percorso avrà due marker in corrispondenza del punto inizio e del punto di fine. I codici del file XML e della classe sono presentati, rispettivamente, nei Listati 4.12 e 4.13.

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"

    android:orientation="vertical"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbarMap"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/background_light"
        app:titleTextColor="@android:color/black"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />


    <org.osmdroid.views.MapView
        android:id="@+id/mapview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>


</LinearLayout>
```

Listato 4.12. Il codice del file XML associato all'activity `MappaActivity`

```
package com.example.marchetrekking;

public class Mappa extends AppCompatActivity {

    RoadManager roadManager;
    private final int PERMISSION = 1;
    private MapView mMapView;
    private IMapController mMapController;
    private String mappa;
    private List<Double> lat = new ArrayList<>();
    private List<Double> lon = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mappa);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbarMap);
        toolbar.setTitle("Mappa");

        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);

        Configuration.getInstance().setUserAgentValue(BuildConfig.APPLICATION_ID);


        if (ActivityCompat.checkSelfPermission(this,
        Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED &&
                ActivityCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
                ActivityCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
            Manifest.permission.ACCESS_FINE_LOCATION, Manifest.permission.ACCESS_COARSE_LOCATION}, PERMISSION);
        }


        mappa = getIntent().getExtras().getString("mappa");
        String[] splitter = mappa.split(",");
        for (int i = 0; i < splitter.length; i++) {
            if (i % 2 == 0) {
                Double d = Double.parseDouble(splitter[i]);
                lat.add(d);
            } else {
                lon.add(Double.parseDouble(splitter[i]));
            }
        }

        setup_map(lat.get(0), lon.get(0));

        List<GeoPoint> geoPoints = new ArrayList<>();
        for (int i = 0; i < lat.size(); i++) {
            geoPoints.add(new GeoPoint(lat.get(i), lon.get(i)));
        }
        Polyline polyline = new Polyline();
        polyline.setPoints(geoPoints);
        mMapView.getOverlays().add(polyline);

        //start
        Marker startMarker = new Marker(mMapView);
        startMarker.setPosition(geoPoints.get(0));
        mMapView.getOverlays().add(startMarker);
        startMarker.setTitle("Start");
        startMarker.setIcon(getResources().getDrawable(R.drawable.marker_default));
        Drawable icon = getResources().getDrawable(R.drawable.pedone);
        startMarker.setImage(icon);

        //end
```

```
            Marker endMarker = new Marker(mMapView);
            endMarker.setPosition(geoPoints.get(geoPoints.size() - 1));
            mMapView.getOverlays().add(endMarker);
            endMarker.setTitle("Fine");
            endMarker.setIcon(getResources().getDrawable(R.drawable.marker_default));
            Drawable icon2 = getResources().getDrawable(R.drawable.pedone);
            endMarker.setImage(icon2);

            insert_scale();

        }

        private void setup_map(double lat, double lon) {
            mMapView = (MapView) findViewById(R.id.mapview);
            mMapView.setTileSource(TileSourceFactory.MAPNIK);
            mMapView.setMultiTouchControls(true);
            mMapView.setBuiltInZoomControls(false);
            mMapController = mMapView.getController();
            mMapController.setZoom(16);

            //dove la mappa viene centrata
            GeoPoint gPt = new GeoPoint(lat, lon);
            mMapController.setCenter(gPt);

        }


        private void insert_scale(){
            ScaleBarOverlay myScaleBarOverlay = new ScaleBarOverlay(mMapView);
            mMapView.getOverlays().add(myScaleBarOverlay);
        }


    }
```

Listato 4.13. Il codice della classe associato all'activity `MappaActivity`

L'activity RecensioneActivity

L'activity `RecensioneActivity` mostra l'elenco delle recensioni di un determinato percorso. Il meccanismo di funzionamento è simile a quello visto per l'activity "elenco percorsi", ma con alcune differenze. In particolare, viene creato un array di recensioni tramite l'utilizzo della classe `DatiRecensioni`. Inoltre, il layout della lista è differente in quanto non compare un'immagine, ma bensi due TextView, che identificano il testo e il nome degli utenti. I codici del file XML e della classe sono presentati, rispettivamente, nei Listati 4.14 e 4.15.

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:id="@+id/recPercorsi"
    tools:context=".Recension">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbarRec"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/background_light"
        app:titleTextColor="@android:color/black"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />

    <ListView
        android:id="@+id/listRec"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        />
</LinearLayout>
```

Listato 4.14. Il codice del file XML associato all'activity `RecensioneActivity`

```
package com.example.marchetrekking;

public class Recension extends AppCompatActivity {

    SessionManager session;
    private LinearLayout ly;
    private ListView lv;
    private ArrayDatiRecensioni listAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_recension);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbarRec) ;
        toolbar.setTitle("Recensioni");

        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        session=new SessionManager(this);
        final HashMap<String,String> utente = session.getUserDetail();

        lv=(ListView) findViewById(R.id.listRec) ;
        ly=(LinearLayout) findViewById(R.id.recPercorsi) ;

        final Intent i=getIntent();

        HttpURLConnection client = null;
        URL url;
        try {
            url = new URL("http://marchetrekking.altervista.org/recensioni.php");

            client = (HttpURLConnection) url.openConnection();
            client.setRequestMethod("POST");
            client.setDoOutput(true);
            client.setDoInput(true);
            OutputStream out = new BufferedOutputStream(client.getOutputStream());
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
            String nome = i.getExtras().getString("NomePercorso");
            String nClean = nome.replace("'", "''");
            String data = URLEncoder.encode("percorso", "UTF-8")
                    + "=" + URLEncoder.encode(nClean , "UTF-8");
            writer.write(data);
            writer.flush();
            writer.close();
            out.close();

            InputStream in = client.getInputStream();
            String json_string = ReadResponse.readStream(in).trim();

            if(!json_string.equals("0[]")) {
                JSONObject json_data = convert2JSON(json_string);
                fill_listview(json_data);
            }
            else{
                Toast.makeText(this, "Nessuna recensione presente", Toast.LENGTH_SHORT).show();
                finish();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        finally{
            if (client!= null){
                client.disconnect();
            }
        }

       lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
           @Override
           public void onItemClick(AdapterView<?> parent, View view, final int position, long id) {
               final int p = position;
               if(listAdapter.getItem(p).getUtente().equals(utente.get(SessionManager.NAME)))
               {
                   AlertDialog.Builder builder1 = new AlertDialog.Builder(Recension.this);
                   builder1.setMessage("Vuoi cancellare questa recensione?");
                   builder1.setCancelable(true);

                   builder1.setPositiveButton(
                           "Yes",
                           new DialogInterface.OnClickListener() {
                               public void onClick(DialogInterface dialog, int id) {
                                   if (listAdapter.getItem(p).getUtente().equals(utente.get(SessionManager.NAME))) {
                                       HttpURLConnection client = null;
                                       URL url;
                                       try {
                                           url = new URL("http://marchetrekking.altervista.org/cancellaRecensione.php");

                                           client = (HttpURLConnection) url.openConnection();
                                           client.setRequestMethod("POST");
                                           client.setDoOutput(true);
                                           client.setDoInput(true);
                                           OutputStream out = new BufferedOutputStream(client.getOutputStream());
                                           BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out,
                                            "UTF-8"));
```

```
                                    String data = URLEncoder.encode("recensione", "UTF-8")+ "=" +
                                            URLEncoder.encode(Integer.toString(listAdapter.getItem(p).getId()),
                                            "UTF-8");

                                    writer.write(data);
                                    writer.flush();
                                    writer.close();
                                    out.close();

                                    InputStream in = client.getInputStream();
                                    String json_string = ReadResponse.readStream(in).trim();

                                    if (!json_string.equals("ok")) {
                                        Toast.makeText(Recension.this, "Errore!", Toast.LENGTH_SHORT).show();

                                    } else {
                                        Toast.makeText(Recension.this, "Recensione cancellata",
                                        Toast.LENGTH_SHORT).show();
                                        finish();
                                    }
                                } catch (IOException e) {
                                    e.printStackTrace();
                                } finally {
                                    if (client != null) {
                                        client.disconnect();
                                    }
                                }
                            } else {
                                Toast.makeText(Recension.this, "Non puoi cancellare questa recensione!",
                                Toast.LENGTH_SHORT).show();
                            }

                        }
                    });

                builder1.setNegativeButton(
                        "No",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                            }
                        });
                AlertDialog alert = builder1.create();
                alert.show();
            }
        }
      });
    }

    private void fill_listview(JSONObject json_data){

        ArrayList<DatiRecensioni> drec=new ArrayList<>();
        Iterator<String> iter = json_data.keys();
        while (iter.hasNext()) {
            String key = iter.next();
            try {
                JSONObject value = json_data.getJSONObject(key);
                //prelevo i dati ottenuti in risposta dal php
                int id = value.getInt("IdRecensione");
                String nome = value.getString("NomePercorso");
                String utente =value.getString("UserName");
                String descrizione =value.getString("Recensione");

                DatiRecensioni dr=new DatiRecensioni(id, nome, descrizione, utente);
                //aggiunta dell'oggetto all'arraylist
                drec.add(dr);
            } catch (JSONException e) {
                // Something went wrong!
            }
        }
        listAdapter = new ArrayDatiRecensioni(Recension.this, drec);
        lv.setAdapter(listAdapter);
    }

    private JSONObject convert2JSON(String json_data){
        JSONObject obj = null;
        try {
            obj = new JSONObject(json_data);
        } catch (Throwable t) {
        }
        return obj;
    }
}
```

Listato 4.15. Il codice della classe associato all'activity `RecensioneActivity`

L'activity AggiungiRecensioneActivity

L'activity `AggiungiRecensioneActivity` mostra la sezione dell'app dedicata alle recensioni che gli utenti possono lasciare sui percorsi. L'utente deve selezionare, sull'apposito Spinner, il percorso che intende recensire e, poi, nello spazio a disposizione, scrivere un breve testo. Quando verrà cliccato il pulsante "Aggiungi", la recensione verrà inviata e salvata nel database assieme allo username dell'utente. I codici del file XML e della classe sono presentati, rispettivamente, nei Listati 4.16 e 4.17.

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/parent_linear_layout2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".add_recension">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbarRec"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/background_light"
        app:titleTextColor="@android:color/black"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />

    <TextView
        android:id="@+id/textView10"
        android:layout_width="380dp"
        android:layout_height="37dp"
        android:text="Seleziona␣un␣percorso␣da␣recensire"
        tools:layout_editor_absoluteX="16dp"
        tools:layout_editor_absoluteY="86dp"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"/>

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="345dp"
        android:layout_height="41dp"
        tools:layout_editor_absoluteX="19dp"
        tools:layout_editor_absoluteY="145dp" />

    <TextView
        android:id="@+id/textView13"
        android:layout_width="380dp"
        android:layout_height="37dp"
        android:layout_marginTop="30dp"
        android:text="Scrivi␣la␣recensione"
        tools:layout_editor_absoluteX="19dp"
        tools:layout_editor_absoluteY="234dp"
        android:layout_marginLeft="10dp"/>

    <EditText
        android:id="@+id/editTextR"
        android:layout_width="335dp"
        android:layout_height="304dp"
        android:layout_marginLeft="10dp"
        android:ems="10"
        android:inputType="textMultiLine|textPersonName"
        tools:layout_editor_absoluteX="19dp"
        tools:layout_editor_absoluteY="292dp" />

    <Button
        android:id="@+id/button2"
        android:layout_width="340dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="10dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="10dp"
        android:background="@drawable/rounded_button"
        android:text="AGGIUNGI"
        android:textColor="@android:color/white"/>
</LinearLayout>
```

Listato 4.16. Il codice del file XML associato all'activity `AggiungiRecensioneActivity`

```
package com.example.marchetrekking;

public class add_recension extends AppCompatActivity {

    private Toolbar t;
    SessionManager session;

    TextView textView1;
    TextView textView2;
    Spinner spinner;
    Button button;
    EditText editText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_recension);
        t=(Toolbar) findViewById(R.id.toolbarRec);
        setSupportActionBar(t);
        getSupportActionBar().setTitle("Recensione");

        textView1 = (TextView) findViewById(R.id.textView10);
        textView2 = (TextView) findViewById(R.id.textView13);
        spinner = (Spinner) findViewById(R.id.spinner);
        button = (Button) findViewById(R.id.button2);
        editText = (EditText) findViewById(R.id.editTextR) ;

        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);

        HttpURLConnection client = null;
        try {
            URL url = new URL("http://marchetrekking.altervista.org/percorsi.php");
            client = (HttpURLConnection) url.openConnection();
            client.setRequestMethod("GET");
            InputStream in = client.getInputStream();
            String json_string = ReadResponse.readStream(in);
            JSONObject json_data = convert2JSON(json_string);
            fill_spinner(json_data);
        } catch (IOException e) {
            e.printStackTrace();
        }
        finally{
            if (client!= null){
                client.disconnect();
            }
        }

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String percorso = spinner.getSelectedItem().toString();
                String nomepClean = percorso.replace("'", "''");
                String recensione = editText.getText().toString();

                String recensionClean = recensione.replace("'", "''");
                session=new SessionManager(add_recension.this);
                HashMap<String,String> utente = session.getUserDetail();
                String user = utente.get(SessionManager.NAME);


                if(!isNullOrEmpty(percorso) && !isNullOrEmpty(recensione)){
                    HttpURLConnection client = null;
                    URL url;
                    try {
                        // se la richiesta è POST
                        url = new URL("http://marchetrekking.altervista.org/aggiungi_recensione.php");
                        client = (HttpURLConnection) url.openConnection();
                        client.setRequestMethod("POST");
                        client.setDoOutput(true);
                        client.setDoInput(true);
                        // write data in request
                        OutputStream out = new BufferedOutputStream(client.getOutputStream());
                        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
                        String data = URLEncoder.encode("percorso", "UTF-8")
                                + "=" + URLEncoder.encode(nomepClean, "UTF-8");

                        data += "&" + URLEncoder.encode("recensione", "UTF-8") + "="
                                + URLEncoder.encode(recensionClean, "UTF-8");

                        data += "&" + URLEncoder.encode("user", "UTF-8") + "="
                                + URLEncoder.encode(user, "UTF-8");


                        writer.write(data);
                        writer.flush();
                        writer.close();
                        out.close();

                        InputStream in = client.getInputStream();
                        String json_string = ReadResponse.readStream(in).trim();

                        if (json_string.equals("1")) {
```

```
                        Toast.makeText(add_recension.this, "Recensione␣Aggiunta", Toast.LENGTH_LONG).show();
                        Intent intent = new Intent(add_recension.this, Home.class);
                        startActivity(intent);
                    } else {
                        Toast.makeText(add_recension.this, "Errore␣nell'inserimento", Toast.LENGTH_LONG).show();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    if (client != null) {
                        client.disconnect();
                    }
                }

            }else{
                Toast.makeText(add_recension.this, "Completa␣i␣campi",Toast.LENGTH_SHORT).show();
            }
        }

    });


}

private void fill_spinner(JSONObject json_data){
    ArrayList<DatiPercorsi> dperc=new ArrayList<>();
    Iterator<String> iter = json_data.keys();
    while (iter.hasNext()) {
        String key = iter.next();
        try {
            JSONObject value = json_data.getJSONObject(key);
            //prelevo i dati ottenuti in risposta dal php
            String nome =value.getString("Nome");

            DatiPercorsi dp=new DatiPercorsi(nome);
            dperc.add(dp);

        } catch (JSONException e) {
            // Something went wrong!
        }
    }

    ArrayList<String> nomi = new ArrayList<>();
    ArrayAdapter<String> arrayAd;
    for(int i = 0; i < dperc.size(); i++){
        nomi.add(dperc.get(i).getNome());
    }

    arrayAd = new ArrayAdapter<String>(add_recension.this, android.R.layout.simple_spinner_item, nomi);
    arrayAd.setDropDownViewResource(R.layout.support_simple_spinner_dropdown_item);
    spinner.setAdapter(arrayAd);

}

private JSONObject convert2JSON(String json_data){
    JSONObject obj = null;
    try {
        obj = new JSONObject(json_data);
        //Log.d("My App", obj.toString());
    } catch (Throwable t) {
        //Log.e("My App", "Could not parse malformed JSON: \"" + json_data + "\"");
    }
    return obj;
}

public boolean isNullOrEmpty(String s){
    return s == null || s.isEmpty();
}

}
```

Listato 4.17. Il codice della classe associato all'activity `AggiungiRecensioneActivity`

L'activity NuovoPercorsoActivity

L'activity `NuovoPercorsoActivity` implementa l'inserimento di un nuovo percorso. Dopo aver aggiunto le varie tappe che compongono il percorso, l'utente deve completare i campi, come il nome, la descrizione, la lunghezza, il livello di difficoltà e la durata del percorso. Una volta creato, potrà anche visionarlo dalla sezione "miei percorsi". I codici del file XML e della classe sono presentati, rispettivamente, nei Listati 4.18 e 4.19.

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".NuovoPercorsoDesc"
    android:orientation="vertical">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/background_light"
        app:titleTextColor="@android:color/black"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />

    <android.support.v4.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fillViewport="true">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:scrollX="-25dp">

            <TextView
                android:id="@+id/textView6"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Nome␣del␣percorso" />

            <EditText
                android:id="@+id/nomeP"
                android:layout_width="310dp"
                android:layout_height="wrap_content"
                android:layout_marginTop="13dp"
                android:ems="10"
                android:inputType="textPersonName" />

            <TextView
                android:id="@+id/textView7"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Descrizione␣del␣percorso" />

            <EditText
                android:id="@+id/descP"
                android:layout_width="310dp"
                android:layout_height="wrap_content"
                android:layout_marginTop="13dp"
                android:ems="10"
                android:inputType="textMultiLine|textPersonName" />

            <TextView
                android:id="@+id/textView14"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Lunghezza␣percorso␣(km)" />

            <EditText
                android:id="@+id/lungP"
                android:layout_width="310dp"
                android:layout_height="wrap_content"
                android:layout_marginTop="19dp"
                android:ems="10"
                android:inputType="numberDecimal"
                />

            <TextView
                android:id="@+id/textView15"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Livello␣di␣difficoltà␣(1-5)" />

            <EditText
                android:id="@+id/livP"
                android:layout_width="310dp"
                android:layout_height="wrap_content"
                android:layout_marginTop="31dp"
                android:inputType="number"
                android:ems="10" />

            <TextView
                android:id="@+id/textView16"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Durata␣del␣pecorso␣(ore)" />

            <EditText
```

```
            android:id="@+id/durataP"
            android:layout_width="310dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="23dp"
            android:ems="10"
            android:inputType="numberDecimal"
            />


        <Button
            android:id="@+id/nuovop"
            android:layout_width="310dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="26dp"
            android:background="@drawable/rounded_button"
            android:text="Crea␣nuovo␣percorso"
            android:textColor="@android:color/background_light" />

    </LinearLayout>
</android.support.v4.widget.NestedScrollView>

</LinearLayout>
```

Listato 4.18. Il codice del file XML associato all'activity `NuovoPercorsoActivity`

```
package com.example.marchetrekking;

public class NuovoPercorsoDesc extends AppCompatActivity {
    EditText nomeP,descP,lungP,livP,durataP;
    SessionManager session;
    HashMap<String, String> hashMap;
    String map="";
    private ArrayList<String> latitudine, longitudine;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_nuova_descrizione_percorso);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar2) ;
        toolbar.setTitle("Crea␣nuovo␣percorso");
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        Intent intent = getIntent();
        latitudine= (ArrayList<String>) intent.getSerializableExtra("latitudine");

        longitudine= (ArrayList<String>) intent.getSerializableExtra("longitudine");

        for (int i = 0; i < latitudine.size() - 1; i++) {

               map += latitudine.get(i)+ "," + longitudine.get(i) + ",";
        }
        map += latitudine.get(latitudine.size()-1)+ "," + longitudine.get(latitudine.size()-1);

        nomeP=findViewById(R.id.nomeP);
        descP =findViewById(R.id.descP);
        lungP=findViewById(R.id.lungP);
        livP=findViewById(R.id.livP);
        durataP=findViewById(R.id.durataP);
        Button crea = findViewById(R.id.nuovop);

        session=new SessionManager(this);

        crea.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                HashMap<String,String> utente = session.getUserDetail();
                String n = utente.get(SessionManager.NAME);
                String nP = nomeP.getText().toString();
                String nomepClean = nP.replace("'", "''");
                String dP = descP.getText().toString();
                String descClean = dP.replace("'", "''");
                String luP = lungP.getText().toString();
                String liP = livP.getText().toString();

                String duP = durataP.getText().toString();

                if(!isNullOrEmpty(n) && !isNullOrEmpty(nP) && !isNullOrEmpty(dP) &&
                 !isNullOrEmpty(luP) && !isNullOrEmpty(liP) && !isNullOrEmpty(duP)) {

                   if(Integer.parseInt(liP) < 6 && Integer.parseInt(liP) > 0){
                       HttpURLConnection client = null;
                       URL url;
                       try {
                           // se la richiesta è POST
                           url = new URL("http://marchetrekking.altervista.org/aggiungi_percorso.php");
                           client = (HttpURLConnection) url.openConnection();
                           client.setRequestMethod("POST");
                           client.setDoOutput(true);
                           client.setDoInput(true);
```

```
                        // write data in request
                        OutputStream out = new BufferedOutputStream(client.getOutputStream());
                        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
                        String data = URLEncoder.encode("nome", "UTF-8")
                                + "=" + URLEncoder.encode(nP, "UTF-8");

                        data += "&" + URLEncoder.encode("desc", "UTF-8") + "="
                                + URLEncoder.encode(dP, "UTF-8");


                        data += "&" + URLEncoder.encode("lunghezza", "UTF-8") + "="
                                + URLEncoder.encode(luP, "UTF-8");


                        data += "&" + URLEncoder.encode("livello", "UTF-8") + "="
                                + URLEncoder.encode(liP, "UTF-8");


                        data += "&" + URLEncoder.encode("durata", "UTF-8") + "="
                                + URLEncoder.encode(duP, "UTF-8");


                        data += "&" + URLEncoder.encode("map", "UTF-8") + "="
                                + URLEncoder.encode(map, "UTF-8");


                        data += "&" + URLEncoder.encode("user", "UTF-8") + "="
                                + URLEncoder.encode(n, "UTF-8");

                        writer.write(data);
                        writer.flush();
                        writer.close();
                        out.close();

                        InputStream in = client.getInputStream();
                        String json_string = ReadResponse.readStream(in).trim();

                        if (json_string.equals("1")) {
                            Intent intent = new Intent(NuovoPercorsoDesc.this, Home.class);
                            startActivity(intent);
                        } else {
                            Toast.makeText(NuovoPercorsoDesc.this, "Errore␣nell'inserimento",
                            Toast.LENGTH_LONG).show();
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    } finally {
                        if (client != null) {
                            client.disconnect();
                        }
                    }
                }else{
                    Toast.makeText(NuovoPercorsoDesc.this, "Livello␣difficoltà␣non␣corretto",
                    Toast.LENGTH_SHORT).show();
                }



            }else{
                Toast.makeText(NuovoPercorsoDesc.this, "Completa␣i␣campi",Toast.LENGTH_SHORT).show();

            }
        }
    });
    }

    public boolean isNullOrEmpty(String s){
        return s == null || s.isEmpty();
    }
}
```

Listato 4.19. Il codice della classe associato all'activity `NuovoPercorsoActivity`

4.3 Manuale utente

Dopo aver installato l'app sul dispositivo, l'utente potrà accedere ad essa premendo sull'icona, presente nel menu dello smartphone (Figura 4.2).

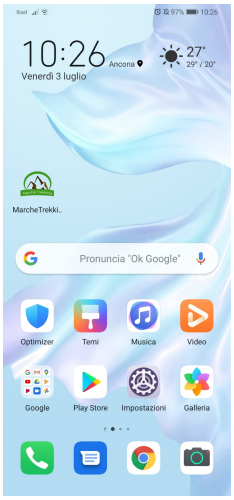

Figura 4.2. Icona visibile dal menu del dispositivo.

In seguito all'apertura, all'utente viene presentata la schermata principale dell'applicazione (Figura 4.3), nella quale le operazioni possibili sono due, ovvero registrarsi oppure accedere alle funzionalità.

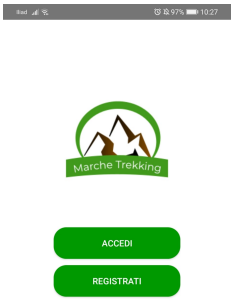

Figura 4.3. Schermata principale mostrata all'avvio dell'app.

Se l'utente è gia registrato, allora potrà procedere direttamente con la fase di login digitando le proprie credenziali nelle rispettive caselle di testo, come mostra la Figura 4.4.

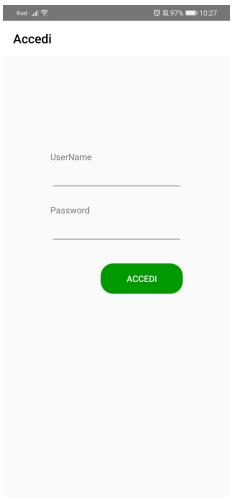

Figura 4.4. Login dell'app.

Invece, se non è registrato, dovrà provvedere a farlo. Nella Figura 4.5 vengono mostrati tutti i campi (nome, cognome, email, username, telefono, password e data di nascita) che devono essere obbligatoriamente completati.

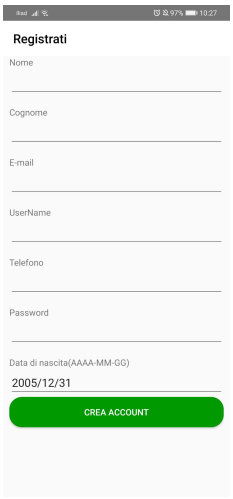

Figura 4.5. Lista dei campi da completare per la registrazione di un utente.

Dopo essersi registrato ed aver effettuato il login, l'utente sarà indirizzato alla home page dell'applicazione (Figura 4.6). Qui, i pulsanti presenti garantiscono l'accesso a tutte le funzionalità messe a disposizione, come, ad esempio, l'elenco dei percorsi, l'inserimento di un percorso o l'inserimento di una recensione, etc. Infine, in alto sulla toolbar, sono visibili le icone per il logout e per le chiamate di emergenza.

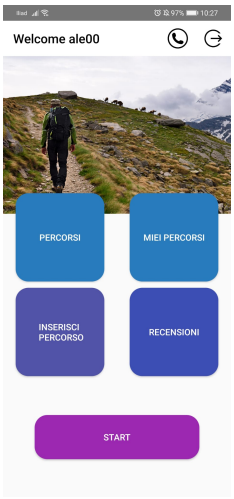

Figura 4.6. Home dell'app.

Quando l'utente aziona il tasto "percorsi" visualizzerà l'elenco dei percorsi presenti all'interno dell'app (Figura 4.7). Per ogni elemento della lista, si può leggere il nome dell'itinerario e pochissime righe della sua descrizione; quest'ultima viene ampliata nella sezione del dettaglio di un percorso.

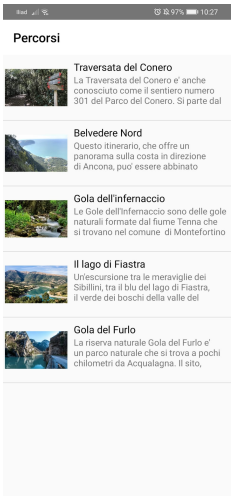

Figura 4.7. Elenco dei percorsi disponibili all'interno dell'app.

Scegliendo un elemento della lista, l'utente aprirà la schermata "dettaglio percorso", inerente al percorso selezionato (Figura 4.8). Oltre alla descrizione dettagliata, è possibile visionare la mappa, le recensioni, le attrazioni ed i prodotti.

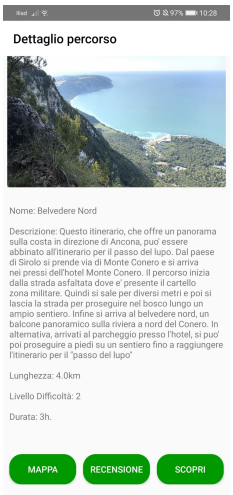

Figura 4.8. Dettagli di un percorso.

La Figura 4.9 rappresenta la mappa associata al percorso scelto in precedenza. Si possono vedere i punti di inizio e di fine del percorso, assieme al tracciato che lo caratterizza.

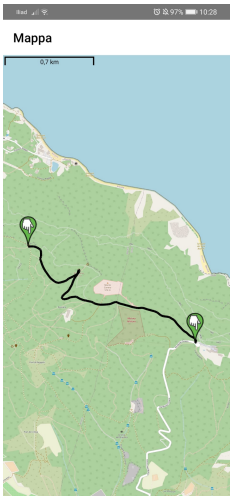

Figura 4.9. Mappa di un percorso.

Con il secondo pulsante della schermata “dettaglio percorso”, l’utente visualizzerà l’elenco delle recensioni disponibili per il percorso selezionato (Figura 4.10).

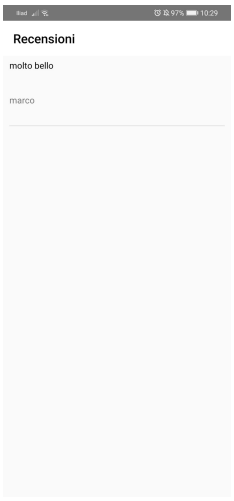

Figura 4.10. Elenco delle recensioni di un percorso.

Invece, tramite il tasto "scopri", egli verrà indirizzato alla schermata dove può scegliere di vedere sia le attrazioni relative al percorso, che i prodotti enogastronomici tipici di esso (Figura 4.11).

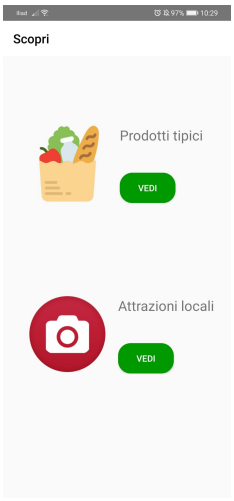

Figura 4.11. Attrazioni relative ad un percorso.

Con il primo tasto "vedi", l'utente visualizza l'elenco dei prodotti tipici legati al percorso attuale (Figura 4.12).

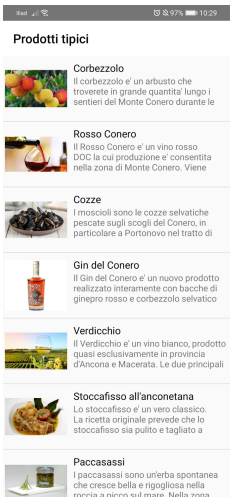

Figura 4.12. Lista di prodotti relativi ad un percorso.

Invece, attraverso il secondo tasto "vedi", l'utente viene indirizzato alla schermata che mostra l'elenco dei punti di attrazione presenti nell'itinerario (Figura 4.13).

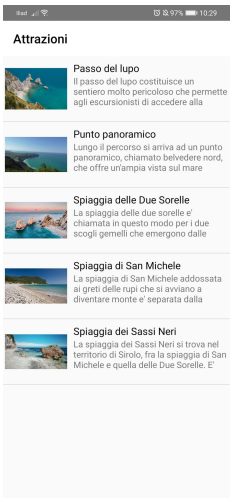

Figura 4.13. Lista delle attrazioni presenti in un percorso.

Selezionando un elemento dalla lista dei prodotti tipici di un percorso, l'utente accederà al dettaglio di un prodotto (Figura 4.14). La stessa identica cosa accade nel momento in cui l'utente seleziona un'attrazione.

Figura 4.14. Dettaglio di un prodotto relativo ad un percorso.

Dopo essere tornato alla home page, l'utente può decidere di inserire un nuovo percorso; in tal caso, esso viene indirizzato alla schermata rappresentata dalla Figura 4.15. Tramite il tasto "aggiungi" è possibile inserire il numero di tappe che compongono il percorso; mentre con "undo", viene annullato, di volta in volta, l'ultimo inserimento della tappa.

Figura 4.15. Inserimento di un nuovo percorso.

Azionando il pulsante "continua" dell'immagine precedente, l'utente procederà con la creazione di un nuovo percorso. Nella Figura 4.16 vengono mostrati i campi che devono essere necessariamente compilati. Alla fine, l'operazione può essere conclusa tramite il pulsante "crea nuovo percorso".

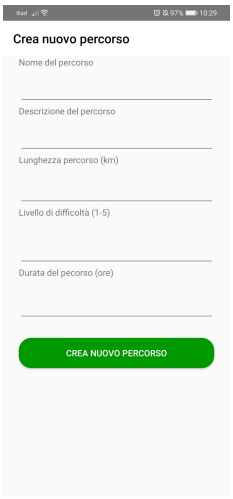

Figura 4.16. Creazione di un nuovo percorso.

Nella Figura 4.17 è mostrata la sezione “miei percorsi”, accessibile direttamente dalla home page dell’app. In essa, sono presenti tutti i percorsi creati dall’utente.

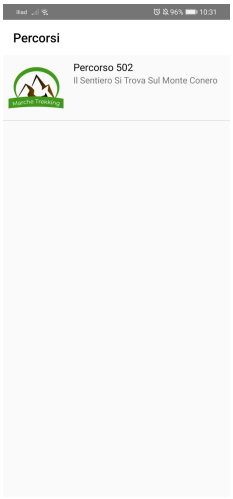

Figura 4.17. Percorsi creati dall’utente.

Per mezzo del pulsante “recensioni”, presente nella home page, l’utente viene rimandato alla schermata che consente di aggiungere una recensione (Figura 4.18). L’aggiunta di una recensione avviene selezionando prima il percorso da recensire e, poi, scrivendo la recensione vera e propria.

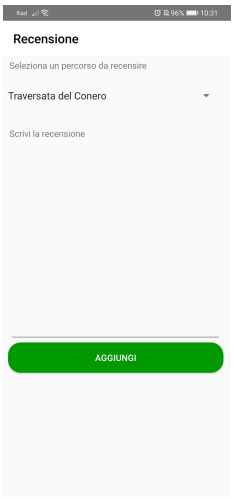

Figura 4.18. Inserimento di una recensione.

La Figura 4.19 rappresenta i numeri da chiamare in caso di eventuali emergenze. Cliccando sul simbolo della cornetta telefonica si viene immediatamente indirizzati al tastierino del telefono con il numero di emergenza già visibile.

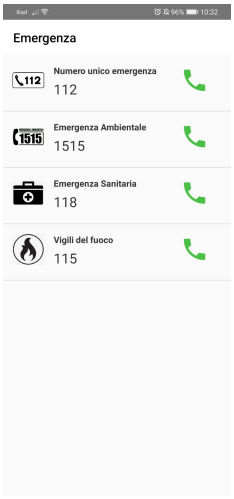

Figura 4.19. Elenco numeri di emergenza.

5

Discussioni in merito al lavoro svolto

Questo capitolo si compone di due sezioni: nella prima si farà una valutazione dell'esperienza avuta e verrà proposta una panoramica sulle nozioni apprese durante lo sviluppo del progetto; nella seconda, invece, si presenterà Open Street Map che svolge un'importante funzione nel sistema realizzato.

5.1 Nozioni apprese sull'utilizzo di Android Studio

Prima di tutto, è di assoluta importanza dire che la realizzazione di questo progetto ha permesso l'approfondimento dei concetti basilari relativi alla progettazione di un sistema. Partendo dall'analisi dei requisiti fino ad arrivare alle fasi di implemetazione e realizzazione, questo lavoro è stato molto utile per immergersi in una situazione reale e comprendere, anche se in minima parte, come si esegue lo sviluppo di un progetto all'interno di un'azienda.

Attraverso l'utilizzo dell'IDE Android Studio si sono consolidate conoscenze fondamentali non solo, ovviamente, per lo sviluppo di applicazioni native, ma anche per la programmazione ad oggetti, un paradigma estremamente importante nell'informatica. Tra le nozioni apprese, oltre i vari componenti di Android con le quali è stata realizzata l'app, troviamo:

- SharedPreferences;
- HTTPUrlConnection;
- AlertDialog;
- SensorManager;
- Intent;
- DatePicker;
- Layout.
- Gradle Script.

Nel seguito daremo uno sguardo a ciascuna di queste nozioni.

SharedPreferences

SharedPreferences è una metodologia utilizzabile per il salvataggio permanente dei dati. La sua particolarità è quella di possedere una struttura chiave/valore. I dati

collocati nelle SharedPreferences vengono salvati in un file XML contenuto nello storage interno. Pertanto si potrà accedere alle Preferences in due modi; ovvero nella modalità di default con il metodo `getPreferences()`, oppure specificando un nome di file con `getSharedPreferences(String filename, int mode)`, dove il primo parametro indica il nome che si vuole dare al file di preferenze, mentre il secondo denota i privilegi da concedere al file.

HTTPUrlConnection

Essa è una connessione URL con supporto per funzionalità specifiche HTTP. Gli usi di questa classe seguono il seguente schema:

- Ottenere una nuova `HttpURLConnection` richiamando `openConnection()` sull'URL scelto.
- Preparare la richiesta. La proprietà principale di una richiesta è il suo URI. Le intestazioni delle richieste possono includere anche metadati.
- Caricare il corpo della richiesta.
- Leggere la risposta. Il corpo della risposta può essere letto dallo stream restituito da `URLConnection.getInputStream()`. Se la risposta non ha un corpo, quel metodo restituisce un flusso vuoto.
- Dopo aver letto il corpo della risposta, `HttpURLConnection` deve essere chiusa chiamando il metodo `disconnect()`. La disconnessione rilascia le risorse detenute da una connessione in modo che possano essere chiuse o riutilizzate.

AlertDialog

Il Dialog è una piccola finestra che permette all'utente di prendere una decisione o aggiungere informazioni addizionali. Nell'app viene utilizzato l'AlertDialog il quale può contenere un titolo, un massimo di tre pulsanti, una lista di elementi selezionabili e può avere un layout apposito. Con l'AlertDialog, possiamo, ad esempio, mettere l'utente di fronte ad una scelta positiva, neutra o negativa. Infatti, nel nostro caso, questo componente viene usato per confermare la cancellazione di un percorso creato dall'utente. I Dialog possono essere personalizzati; infatti, si possono definire dei layout in funzione della situazione da gestire.

SensorManager

SensorManager consente di accedere ai sensori del dispositivo. Esso viene utilizzato per la sezione Attività dell'app, dove si implementa un semplice contapassi. La prassi comune per ricavare dati periodici da un sensore è registrare un listener nella nostra applicazione. Ciò comporta l'implementazione di un metodo di callback all'interno del quale si potrà fare uso delle misurazioni rilevate.

Il metodo `onSensorChanged` costituisce il cuore dell'interazione con il sensore. È qui che arrivano le chiamate del listener ogni volta che sono disponibili nuove misurazioni. L'evento notificato verrà formalizzato con un oggetto della classe `SensorEvent`. Quest'ultima permette di leggere i valori recuperati come un array numerico. Questi valori, però, non sono precisissimi; infatti la difficoltà sta proprio nella loro interpretazione.

Intent

Gli Intent vengono utilizzati sia per richiamare una particolare classe nell'applicazione che per richiamare una qualsiasi applicazione che deve lavorare su specifici dati. Nella nostra applicazione sono usati per il passaggio da un'activity all'altra. Il costruttore Intent riceverà due argomenti: il primo è l'azione da compiere, mentre il secondo l'entità in gioco.

DatePicker

In Android, DatePicker è un controllo che consente agli utenti di selezionare la data opzionando giorno, mese e anno. Se utilizziamo DatePicker nella nostra applicazione, ci assicureremo che gli utenti selezionino una data avente un formato valido. Nel caso cui l'applicazione richieda un range di età, si deve controllare che non vengano selezionate date non ammesse. Questa funzionalità viene implementata nella sezione Registrati dell'app quando l'utente deve inserire la propria data di nascita.

Layout

I layout definiscono la struttura dell'interfaccia utente dell'app. Tutti gli elementi nel layout sono costruiti utilizzando una gerarchia formata dagli oggetti View e ViewGroup. View disegna qualcosa sullo schermo con cui l'utente può interagire. Alcune sottoclassi sono: Button e TextView. ViewGroup è un contenitore invisibile che definisce la struttura del layout per oggetti View e ViewGroup. Alcune sottoclassi di ViewGroup sono LinearLayout e ConstraintLayout. Si possono definire i layout in due modi, ovvero dichiarando gli elementi nel file XML, oppure, istanziando gli elementi a runtime. Ogni file di layout deve contenere esattamente un elemento di root, che può essere un oggetto View o ViewGroup. Dopo aver definito l'elemento di root, si possono aggiungere altri oggetti, creando, così, una gerarchia e definendo il layout.

Gradle Script

Gradle Script contiene tutti i file di configurazione di build del progetto. Android Studio utilizza Gradle come strumento avanzato di building, per automatizzare e gestire il processo di costruzione dell'applicazione, mantenendo un'ottima flessibilità. I vari componenti del Gradle sono:

- `settings.gradle`: definisce quali moduli caricare quando l'app viene costruita;
- `build.gradle` (Module): definisce la configurazione di build che si applicherà a tutti i moduli del progetto;
- `build.gradle` (Project): definisce la configurazione di build per il singolo modulo;
- `gradle.properties`: sono le impostazioni del Gradle;
- `local.properties`: sono le proprietà dell'ambiente locale per il processo di building, come il path di installazione delle SDK.

5.2 Open Street Map

Open Street Map nel seguito (OSM) è un progetto finalizzato a creare mappe del mondo liberamente accessibili. La caratteristica fondamentale è che i dati in OSM hanno una licenza libera, quindi possono essere utilizzati per qualsiasi scopo, con l'unico vincolo di citare la fonte ed usare la stessa licenza per i lavori derivati da essi. Le mappe sono state create usando come riferimento i dati forniti da dispositivi GPS portatili, fotografie aeree ed altre fonti libere. Vengono, anche, effettuati dei rilievi sul territorio da parte di volontari. Infine, le informazioni raccolte vengono inviate ad un database in formato vettoriale. Il database di Open Street Map è pubblicato secondo la licenza ODbL (Open Database License).

In Android, OSM fornisce una libreria chiamata `osmdroid`. Come primo passo, per utilizzare le mappe di OSM nell'app, bisogna includere due file JAR all'interno del nostro progetto. Successivamente, per aggiungere le dipendenze dei due file `.jar` nel progetto, è necessario definire i valori in alcune righe nel Gradle. Nell' `AndroidManifest.xml` vanno inseriti i permessi essenziali per l'utilizzo di OSM. Solo a questo punto si può aggiungere un MapView all'XML layout (Listato 5.1) e utilizzare l'oggetto MapView nella relativa activity (Listato 5.2).

```
<org.osmdroid.views.MapView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/mapview">
</org.osmdroid.views.MapView>
```

Listato 5.1. Il codice dell'oggetto MapView in un XML layout

```
private MapView mapView;
private MapController mapController;
mapView = (MapView) this.findViewById(R.id.mapview);
mapView.setBuiltInZoomControls(true);
mapView.setMultiTouchControls(true);
mapController = this.mapView.getController();
mapController.setZoom(2);
```

Listato 5.2. Il codice dell'oggetto MapView in un'activity

Sia `MapView` che `MapController` sono classi di `osmdroid`. La prima estende ViewGroup, e consiste in una vista per mostrare una mappa. La seconda serve per gestire la mappa di MapView, come lo zoom o le animazioni.

5.2.1 Punti di forza

Open Street Map offre diversi vantaggi, tra cui:

- *Aggiornabilità*: essendo modificabile liberamente dagli utenti, il tempo di aggiornamento della mappa è legato alla velocità di caricamento dei nuovi dati nel server.
- *Completezza*: il livello di dettaglio della mappa è molto elevato in quanto è possibile segnalare su di essa diversi oggetti.
- *Adattamento*: ci sono diversi livelli di mappa, i quali includono trasporti, piste ciclabili e mappe umanitarie.

- *Licenza libera*: questo è il punto principale. A qualsiasi individuo viene offerta la possibilità di usare gratuitamente e legalmente le mappe per farne ciò che desidera, a condizione di citare la fonte.

5.2.2 Punti di debolezza

Invece, tra punti di debolezza che caratterizzano Open Street Map troviamo:

- *Grafica*: la grafica non è molto accattivante. Per acquisire confidenza bisognerà navigarci un pò, in modo tale da capire tutti i segni sulla mappa. Inoltre, la maggior parte di questi risultano abbastanza intuitivi.
- *Navigazione*: nonostante possa funzionare da navigatore, OSM non è adatta allo scopo come Google Maps. OSM è intransigente, basta una virgola di troppo e non si riesce a trovare l'indirizzo desiderato. Questo perché, in verità, OSM, più che una mappa, è un database.
- *Completezza*: oltre ad essere un vantaggio, potrebbe essere anche uno svataggio, perchè se nessuno si è mai interessato di una determinata zona, quella avrà inesorabilmente pochi dettagli.

6

Conclusioni e uno sguardo al futuro

In questo elaborato di tesi si è visto il progetto MarcheTrekking, che consiste nella progettazione e implementazione di un'app Android a supporto delle attività di trekking nella Regione Marche.

Prima di tutto è stata presentata una panoramica sul sistema operativo Android, fornendo alcuni dati statistici e illustrandone la struttura e gli strumenti di sviluppo. Successivamente, ci si è concentrati sulle app, spiegando cosa sono e le loro tipologie, facendo particolare attenzione sul concetto di app nativa.

Dal Capitolo 2 in poi ci si è concentrati sull'app oggetto della presente tesi. In particolare, nel Capitolo 2 sono state effettuate una descrizione del contesto e un'analisi dei requisiti, illustrando le funzionalità da implementare e i vincoli da rispettare per poi passare alla fase di progettazione.

Il Capitolo 3 ha trattato la fase di progettazione della base di dati e dell'app; sono stati presentati lo schema E-R e lo schema relazionale, insieme a dei diagrammi UML e dei mockup utili per capire il funzionamento dell'app.

Il Capitolo 4 ha trattato l'implementazione. In esso si è mostrata la struttura dell'app illustrando e spiegando tutti i listati che la compongono. Inoltre, sempre in questo capitolo, è stata presentata una guida all'utilizzo dell'applicazione la quale mostra i diversi passaggi che un utente deve eseguire per poterla utilizzare in modo efficiente ed efficace.

Infine, nel Capitolo 5, è stato proposto un breve riepilogo degli argomenti affrontati e delle nozioni apprese durante la realizzazione del progetto, per poi eseguire un'analisi conclusiva dell'elaborato.

Questo progetto è risultato essere molto utile per comprendere come sia possibile implementare un'applicazione nativa Android, integrarla con le mappe e farla interagire con un database.

L'app, sebbene sia funzionante, può essere migliorata in diversi aspetti. Ad esempio, si potrebbe raffinare l'aspetto grafico, dal momento che per versioni diverse di Android e per dimensioni diverse di smartphone, alcuni elementi grafici non vengono visualizzati nella stessa maniera. Inoltre, si potrebbero introdurre le funzionalità di un GPS che segnalino constantemente la posizione dell'utente, si potrebbe migliorare il registro delle attività e si potrebbe aumentare il numero di percorsi disponibili. Infine, si potrebbe alleggerire l'applicazione gestendo meglio gli elementi grafici, quali foto e immagini varie.

Riferimenti bibliografici

1. Android. https://it.wikipedia.org/wiki/Android, 2020.
2. Android Developer: Activity. https://developer.android.com/reference/android/app/Activity, 2020.
3. Android Developer: Android Studio. https://developer.android.com/studio/intro, 2020.
4. Android Developer: DatePicker. https://developer.android.com/reference/kotlin/androidx/leanback/widget/picker/DatePicker?hl=en, 2020.
5. Android Developer: Dialogs. https://developer.android.com/guide/topics/ui/dialogs?hl=en, 2020.
6. Android Developer: HTTPUrlConnection. https://developer.android.com/reference/java/net/HttpURLConnection, 2020.
7. Android Developer: SharedPreferences. https://developer.android.com/reference/android/content/SharedPreferences, 2020.
8. Android: Origini, storia, versioni varianti del robottino verde. https://www.tuttoandroid.net/android/, 2020.
9. Applicazione Mobile. https://it.wikipedia.org/wiki/Applicazione_mobile, 2020.
10. Introduzione ad Android. https://www.apogeonline.com/contrib/uploads/capitolo1-a9.pdf, 2020.
11. Open Street Map. https://it.wikipedia.org/wiki/OpenStreetMap, 2020.
12. Open Street Map. https://www.openstreetmap.org, 2020.
13. Smartphone. https://it.wikipedia.org/wiki/Smartphone, 2020.
14. M. Canducci. *XML. Conoscere il linguaggio XML significa poter comunicare veramente con tutti*. Apogeo, 2009.
15. M. Carli. *Android 6. Guida per lo sviluppatore*. Apogeo, 2016.
16. C. De Sio Cesari. *Manuale di Java 8: Programmazione orientata agli oggetti con Java standard edition 8*. Hoepli, 2014.
17. E. Cisotti and M. Giannino. *Android. Guida completa*. Edizioni LSWR, 2015.
18. F. Collini, M. Bonifazi, A. Martellucci, and S. Sanna. *Android: Programmazione avanzata*. Edizioni LSWR, 2015.
19. P. Deitel, H. Deitel, A. Deitel, and M. Morgano. *Sviluppare app per Android*. Pearson Education Italia, 2012.
20. J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley. *The Java Language Specification, Java SE 8 Edition (Java Series)*. Addison-Wesley Professional, 2014.
21. T. Hagos. *Learn Android Studio 3: Efficient Android App Development* . Apress, 2018.
22. C. Horstmann. *Concetti di Informatica e fondamenti di Java (IV Edizione)*. Apogeo S.R.L., 2007.

23. G. Nudelman. *Android Design Patterns: Interaction Design Solutions for Developers*. Wiley, 2013.
24. H. Schildt. *Java. La guida completa*. McGraw-Hill Education, 2012.
25. I. Sommerville. *Ingegneria del software (Decima edizione)*. Pearson, 2017.
26. A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2011.
27. M. Weisfeld. *The Object-Oriented Thought Process*. Addison-Wesley Professional, 2013.
28. B. Cruz Zapata. *Android Studio Application Development*. Packt Publishing, 2013.

Ringraziamenti

Il percorso affrontato in questi tre anni è stato molto difficile, ma lo sarebbe stato ancora di più senza l'aiuto e il sostegno delle persone a me care.

Prima di tutto voglio ringraziare i miei genitori per esserci sempre stati per me, nonostante le difficoltà che abbiamo dovuto affrontare in questi anni. Mi sono sempre stati vicini in tutto, cercando in ogni modo di aiutarmi nei momenti di scoraggiamento (non pochi) e festeggiando i miei successi come se fossero i loro.

Poi ringrazio di cuore tutta la mia famiglia, sia quella che abita a pochi passi da me, sia quella un pò più distante, che si è sempre interessata ai miei progressi, credendo sempre in me, dandomi, così, la motivazione che mi serviva per andare avanti. Inoltre voglio fare un grande ringraziamento a tutti i miei amici, sia quelli che conosco da una vita, sia quelli che ho incontrato in questo viaggio.

Infine, un enorme ringraziamento va al mio relatore, il Prof. Domenico Ursino e al suo assistente Luca Virgili. Li ringrazio sentitamente per avermi insegnato numerose nozioni, per avermi guidato in questa parte finale del mio percorso e per l'incredibile disponibilità avuta nei miei confronti durante l'intero svolgimento del progetto.